Colour Personal Computer

»CPC6128«

GUIDE de L'UTILISATEUR



Introduction

Schneider CPC6128
Ordinateur à unité de disque intégrée

Une progression logique

Encouragés par l'immense succès des systèmes Schneider CPC464 et CPC664 et mûs par la même impulsion novatrice, nous sommes fiers de vous présenter le CPC6128.

Fidèle à notre concept de compatibilité, le CPC6128 fonctionne avec les logiciels de ses deux prédécesseurs et dispose en outre d'une RAM supplémentaire de 64Ko pour exploitation du système CP/M et autres applications utilisateur.

CP/M PLus

Le système CP/M Plus, appelé également CP/M 3.1, offre une grande facilité d'accès à la bibliothèque des logiciels CP/M80. Les 61Ko de TPA (Transient Program Array ou « tableau de programme transitoire ») fournissent à tous les programmes CP/M80 une vaste capacité d'exploitation et tout l'espace requis pour le stockage des données. Bien que compatibles entre elles, les versions CP/M Plus et CP/M 2.2 sont toutes deux disponibles pour vous permettre d'exploiter sans modification les applications développées avec le système CP/M 2.2.

La plupart des logiciels CP/M 2.2 peuvent bénéficier des nombreuses améliorations du CP/MPlus, sans risque de conflit. La fonction d'émulation des terminaux intégrée au CP/M Plus permet d'installer directement des logiciels configurés pour l'exploitation d'écrans aux caractéristiques analogues à celles des terminaux VT52 et Zénith Z19/Z29.

GSX

Grâce au système d'extension graphique GSX fourni avec le CP/M Plus, les programmes d'application peuvent adresser imprimantes, tables traçantes et écrans utilisant des instructions standard; la notion de portabilité des programmes CP/M est ainsi élargie audelà des seuls écrans de texte. Il est possible, en outre, d'interfacer les logiciels bénéficiant du GSX avec un grand nombre d'imprimantes, pour obtenir des copies sur papier de graphiques, diagrammes, etc.

Dr. LOGO

Tandis que LOGO de Digital Research poursuit ses investigations dans le monde de l'enseignement assisté par ordinateur, la capacité accrue du 6128 permet d'étendre le logiciel Dr. LOGO, utilisé précédemment sous CP/M 2.2. Et, bien entendu, les programmes écrits sous CP/M 2.2 jouissent d'une compatibilité ascendante.

Les disquettes

Avec son système de disquettes intégré, le CPC664 a ouvert la voie du futur; le CPC6128 reprend le flambeau: offrant à une clientèle de plus en plus exigeante une informatique à disquettes pour un prix abordable, il réconcilie enfin les performances de l'informatique personnelle avec les coûts des ordinateurs individuels. Les disquettes de tous les systèmes Schneider sont interchangeables, mais les programmes bénéficiant des améliorations du CP/MPlus et du système GSX ne tournent pas, on s'en doute, sur les systèmes dépourvus de ces avantages.
Derrière le CPC6128 s'inscrit, en fait, l'un des noms les plus prestigieux de l'informatique britannique. Par ailleurs, le club des utilisateurs Schneider et son mensuel jouissent d'une solide réputation pour la qualité et la primeur des informations qu'ils diffusent.

Les logiciels

Le CPC6128 accepte tous les logiciels sur disquette des CPC664 et CPC464/DDI1 et la quasi-totalité des logiciels sur cassette du CPC464 (si un lecteur de cassette lui est raccordé, bien sûr). Il offre ainsi à l'utilisateur un choix fantastique de logiciels immédiatement disponibles, issus de la gamme étendue d'AMSOFT, et de produits existant sur le marché.

AMSOFT

Tout commentaire ou suggestion concernant ce produit ou son manuel sera bienvenu...

AMSOFT et Schneider accepteront volontiers vos suggestions à propos de l'ordinateur ou de ce guide

Toute correspondance doit être adressée à :

Schneider
Computer Division

Silvastraße 1
8939 Türkheim 1

Toute maintenance et service après-vente concernant le produit doivent être effectués obligatoirement par des revendeurs AMSOFT agréés. Ni Amsoft, niSchneiderne seront responsables, de quelque façon que ce soit, de toute perte ou dommage causé par une maintenance ou service effectué par des personnes non agréées.

Ce guide est seulement destiné à faciliter l'utilisation du produit par le lecteur et, par conséquent, ni Amsoft, niSchneiderne seront responsables de toute perte ou dommage quelconque qui pourrait résulter de l'utilisation de toutes informations, renseignements, erreurs ou omissions contenus dans ce guide ainsi que de toute utilisation impropre du produit.

Il est recommandé de joindre le talon d'enregistrement de Digital Research.

Première publication 1985

Compilé par Ivor Spital
Auteurs : Roland Perry, Ivor Spital, William Poel, Cliff Lawson, avec accord de Locomotive Software Ltd.

Publié par AMSOFT
Traduit et adapté par le Groupe KUJAWSKI et l'équipe d'AMSOFT, France.

IMPORTANT

Remarques relatives à l'installation

1. Suivez les instructions dans la partie 1 du Cours Elémentaire et utilisez une prise à 3 broches pour le raccordement au secteur.

2. Ne tentez jamais de brancher le système sur une autre tension que 220 V CA, 50 Hz.

3. Aucune pièce du système n'étant dépannable par l'utilisateur, il est superflu et peu recommandé de chercher à y accéder. Pour le dépannage, faites toujours appel à un technicien qualifié.

4. Pour plus de confort visuel, éloignez le moniteur le plus possible du clavier et réglez la luminosité de l'écran en fonction de l'éclairage de la pièce. La commande de brillance du moniteur doit être réglée sur le niveau le plus faible possible.

5. L'ordinateur doit se trouver en face du moniteur et le plus loin possible de ce dernier. Pour une fiabilité maximale, placez l'unité de disque à droite de l'écran. Evitez d'installer l'ordinateur à proximité d'une source d'interférences.

6. Veillez à éloigner les disques et les unités de disque des champs magnétiques.

7. Si vous disposez d'une deuxième unité de disque, séparez le câble plat d'interconnexion de l'unité du câble d'alimentation secteur.

8. Veillez à dégager les orifices de ventilation du système.

9. Evitez d'entreposer votre matériel dans des endroits trop chauds, froids, humides ou générateurs de poussière.

Remarques relatives au fonctionnement

(Ne vous inquiétez pas si les termes techniques employés dans ce chapitre vous paraissent obscurs ; ils s'éclairciront au fur et à mesure que vous avancerez dans le manuel).

1. N'oubliez jamais de retirer la disquette de son unité avant de mettre le système sous ou hors tension. Vous risqueriez de l'endommager et de perdre les programmes et données qu'elle contient.

2. Faites toujours une sauvegarde des disquettes contenant des programmes précieux. C'est le cas notamment de la disquette du système d'exploitation CP/M fournie avec le 6128. En cas de perte ou d'altération accidentelle des données d'une disquette, son remplacement pourrait s'avérer des plus coûteux.

3. Pour éviter l'écrasement accidentel des données stockées sur la disquette du CP/M, veillez à ne pas recouvrir ses ouvertures de protection à l'écriture.

4. Si vous utilisez deux unités de disquettes, l'unité Schneider FD1, par exemple, mettez toujours la deuxième sous tension avant d'allumer l'ordinateur.

5. Ne touchez pas la surface magnétique de la disquette.

6. Ne retirez jamais la disquette en cours de lecture ou d'écriture.

7. Rappelez-vous que le formatage d'une disquette entraîne l'effacement des données qu'elle contient.

8. Les progiciels sur cassettes destinés au système Schneider CPC464 font parfois appel à l'espace mémoire occupé par l'interface interne d'exploitation sur disques et sont donc inutilisables sur le 6128 équipé d'un lecteur de cassettes. Pour les problèmes de comptabilité concernant les logiciels sur cassette, adressez-vous directement à AMSOFT. De toute façon, la plupart des logiciels AMSOFT sont disponibles sur disquettes, compatibles avec le système 6128.

9. Le dépôt légal de votre disquette CP/M (encodée d'une référence) ne permet son utilisation que sur un seul système informatique. Vous n'êtes donc pas autorisé à donner cette disquette à un autre utilisateur. A ce sujet, lisez attentivement le texte de dépôt légal situé à la fin du manuel (Annexe 1).

TABLE DES MATIERES

Chapitre 4
Utilisation des disquettes et des cassettes

Chapitre 5
Les éléments d'AMSDOS et du CP/M

Chapitre 6
Introduction au LOGO

Chapitre 6 suite...

Chapitre 7
Pour information...

Annexes

Chapitre 1
Cours élémentaire

Partie 1 : Mise en marche

L'ordinateur personnel Schneider 6128 peut fonctionner avec les écrans suivants :

1. Schneider GT65, moniteur monochrome vert

2. Schneider CTM644, moniteur couleur

3. Schneider MP2, boîtier d'adaptation et téléviseur couleur muni de prise péritel.

Connexion de l'ordinateur à un moniteur

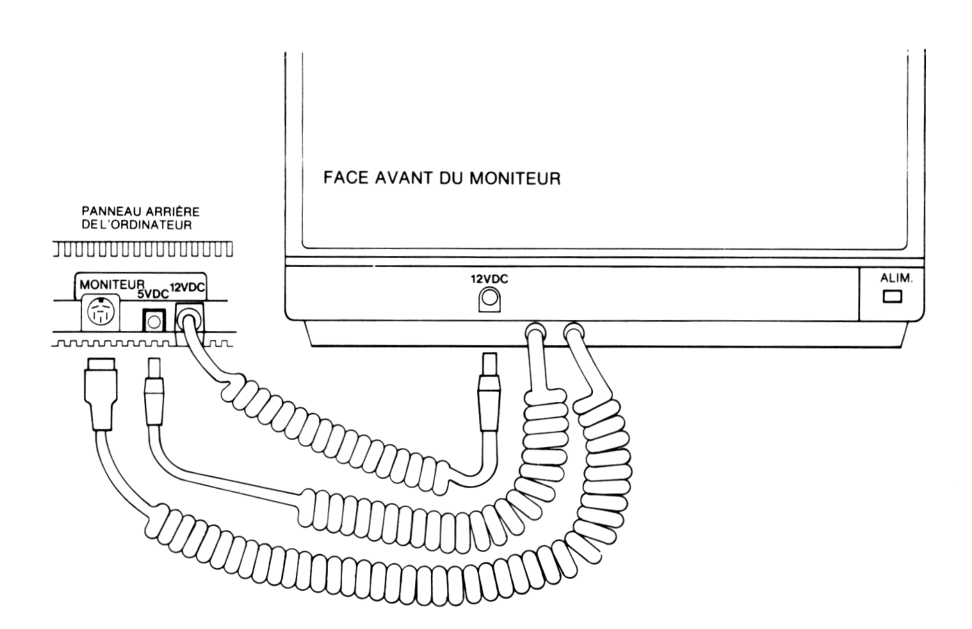
(Si vous utilisez le 6128 avec un boîtier d'adaptation, passez directement au chapitre suivant).

1. Assurez-vous que le moniteur n'est pas branché sur secteur.

2. Connectez le câble sortant de l'avant du moniteur, équipé d'une fiche DIN à 6 broches, à la prise **MONITOR** située à l'arrière de votre ordinateur.

3. Connectez le câble sortant de l'avant du moniteur, équipé d'une fiche **5 V DC**, à la prise **5 V DC**, située à l'arrière de votre ordinateur.

4. Connectez le câble partant de l'arrière de l'ordinateur, équipé d'une petite fiche **12 V CC**, à la prise située à l'avant du moniteur.



Connexion de l'ordinateur à un boîtier d'adaptation MP2

Le MP2 doit se trouver à droite de l'ordinateur.

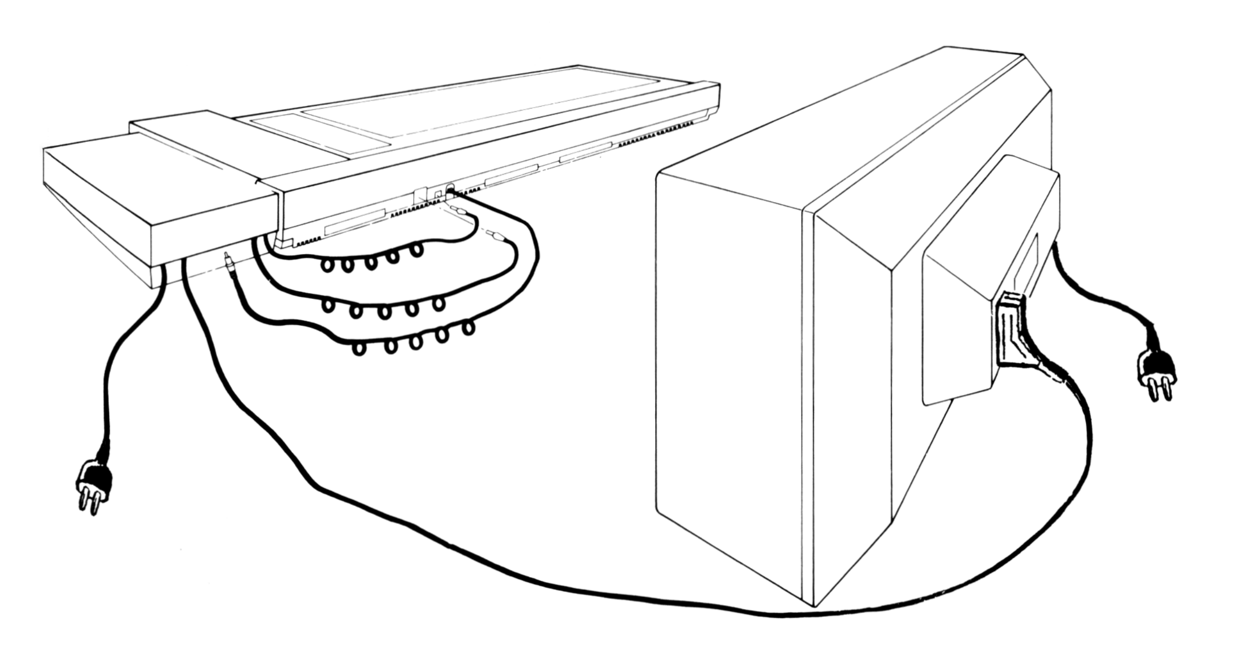
1. Assurez-vous que le MP2 n'est pas branché sur secteur.

2. Connectez la fiche DIN à 6 broches du MP2 à la prise MONITOR située à l'arrière de l'ordinateur.

3. Connectez la fiche 5 V CC du MP2 à la prise 5 V DC située à l'arrière de l'ordinateur.

4. Connectez la prise péritel mâle du MP2 à la prise péritel femelle de votre téléviseur.

5. Connectez la fiche 12 V CC du câble partant de l'arrière de l'ordinateur 12 V DC à la prise située à l'arrière du MP2.



Mise sous tension : système CPC6128 et GT65/CTM644

(Si vous utilisez votre 6128 avec un boîtier d'adaptation MP2, passez directement au chapitre suivant).

Après avoir suivi les instructions de connexion précédentes, insérez la fiche d'alimentation du système dans la prise secteur et effectuez la mise sous tension. Appuyez alors sur le commutateur POWER situé en haut à droite du moniteur lorsque ce commutateur n'est pas enfoncé, le système est hors tension.

Mettez l'ordinateur sous tension à l'aide du commutateur POWER situé à droite.

Le voyant rouge, en haut et au centre du clavier, s'allume tandis que le texte ci-dessous s'affiche sur l'écran du moniteur :

```
 Schneider 128K Microcomputer   (v3)
 C 1985 Amstrad Consumer Electronics plc
            and Locomotive Software Ltd.

 BASIC 1.1

Ready
■
```

Pour éviter de vous fatiguer les yeux inutilement, réglez la commande de brillance (BRIGHTNESS) de façon à obtenir des caractères facilement lisibles sans éblouissement.

Le bouton de réglage de luminosité BRIGHTNESS se trouve à l'avant du GT65 et à droite du CTM644.

Si vous utilisez un moniteur monochrome GT65, vous devez régler le contraste et la hauteur de l'image à l'aide des boutons CONTRAST et VERTICAL HOLD, situés à la partie inférieure de sa face avant.

Le contraste doit être réglé au minimum, tout en permettant une lecture facile des caractères.

Mise sous tension : CPC6128 et boîtier d'adaptation MP2

Après avoir suivi les procédures de connexion précédemment décrites, insérez la fiche d'alimentation du système dans la prise secteur.

Mettez votre téléviseur sous tension.

Mettez l'ordinateur sous tension à l'aide du commutateur POWER situé sur la droite.

Le voyant rouge situé au centre du clavier doit s'allumer tandis que l'écran de votre téléviseur affiche :

```
 Schneider 128K Microcomputer   (v3)
 C 1985 Amstrad Consumer Electronics plc
             and Locomotive Software Ltd.

 BASIC 1.1

Ready
■
```

Réglez votre téléviseur de façon à obtenir une image optimale. Les caractères s'affichent en jaune sur fond bleu foncé.

Autres connexions :

Si vous désirez connecter à votre système d'autres périphériques, tels que :

* manettes de jeu
* lecteur de cassettes
* imprimante
* unité de disque supplémentaire
* enceintes/amplificateur externes
* périphérique(s) d'extension

...vous trouverez de plus amples informations à ce sujet dans la partie 2 du cours élémentaire.

En dernier ressort, vérifiez que vous avez bien respecté les indications mentionnées au début du manuel, dans la rubrique « IMPORTANT » :

REMARQUES RELATIVES A L'INSTALLATION : 1, 2, 3, 4, 5, 6, 7, 8
REMARQUES RELATIVES AU FONCTIONNEMENT : 1

Partie 2 : Connexion de vos périphériques

Cette partie explique comment connecter au système 6128 les divers périphériques ou extensions. Vous trouverez des informations concernant l'utilisation de ces périphériques dans les parties correspondantes du manuel.

Manettes de jeu

La manette de jeu Schneider JY2 est un élément supplémentaire dont vous pourrez, si vous le désirez, enrichir votre ordinateur 6128. Utilisée notamment par certains programmes de jeu, elle sert de manche à balai avec bouton de mise à feu.

Connectez la fiche de la manette à la prise JOYSTICK de votre ordinateur. Vous pouvez utiliser deux manettes ; la seconde sera alors enfichée dans la prise de la première.

Il est également possible d'utiliser la manette de jeu JY1 avec le 6128.

Vous trouverez de plus amples informations sur les manettes de jeu plus loin dans le manuel.

Lecteur/enregistreur de cassettes

Les programmes peuvent être chargés et sauvegardés sur cassettes ou sur disques. Vous trouverez plus loin dans le manuel les commandes gérant ces phériphériques.

Pour connecter un lecteur/enregistreur de cassettes à votre 6128, utilisez le câble AMSOFT CL1 ou un câble d'interconnexion standard équivalent.

Insérez la fiche DIN à 5 broches du câble dans la prise TAPE de l'ordinateur.

Insérez l'extrémité du câble bleu dans la prise REMOTE ou REM (télécommande) de votre lecteur/enregistreur de cassettes.

Insérez l'extrémité du câble rouge dans la prise MIC, COMPUTER IN ou INPUT (entrée enregistrement) de votre lecteur/enregistreur.

Insérez l'extrémité du câble blanc dans la prise EAR (écouteur), COMPUTER OUT (sortie ordinateur) ou OUTPUT (sortie) de votre lecteur/enregistreur.

La qualité de transmission des données entre le 6128 et la cassette dépend largement des niveaux d'enregistrement (LEVEL) et de sortie (VOLUME) de votre lecteur/enregistreur. Si vous ne parvenez pas à charger ou sauvegarder vos programmes correctement, faites varier les réglages jusqu'à obtenir la qualité souhaitée.

Imprimante

Le 6128 peut fonctionner avec toute imprimante parallèle compatible Centronics. Pour l'imprimante Schneider DMP1, utilisez le câble qui l'accompagne.

Si vous utilisez une autre imprimante compatible Centronics, procurez-vous le câble AMSOFT PL1.

Connectez l'extrémité du câble équipée d'un connecteur femelle plat au connecteur mâle PRINTER situé à l'arrière de l'ordinateur.

Connectez ensuite la fiche type Centronics de ce câble dans la prise située à l'arrière de l'imprimante. Si les deux côtés de la prise de l'imprimante sont dotés d'agrafes de fixation, celles-ci doivent s'enclencher dans les ouvertures de part et d'autres de la fiche Schneider

Le fonctionnement de l'imprimante est décrit en détail plus loin dans le manuel.

Unité de disquette supplémentaire (Schneider FD1)

Vous pouvez ajouter au système une seconde unité de disquette, Schneider FD1. Les avantages d'un système à deux unités de disquette apparaîtront nettement aux habitués de CP/M, car de nombreux programmes sont prévus pour exploitation avec un disque programme inséré dans une unité et un autre disque dans une seconde unité pour le stockage des fichiers de données.

L'exploitation d'un programme sous CP/M nécessite toujours son chargement à partir d'une disquette (en effet, le BASIC de la ROM n'est pas accessible). CP/M permet l'exécution de programmes excédant l'espace mémoire disponible grâce à une technique de recouvrement obligeant à découper le programme en plusieurs fichiers. Il est possible que le nombre de fichiers programme contenu sur la disquette soit tel qu'il ne reste plus de place disponible pour les données.

Grâce à la souplesse des utilitaires fournis avec le système 6128 à disquette, vous pouvez effectuer avec une seule unité de disquette toutes les manipulations de fichiers dont vous avez besoin : copie, effacement, etc. Quoi qu'il en soit, une seconde unité accélérera ces opérations et réduira les risques de mauvaises manipulations.

Pour connecter l'unité FD1 au 6128, utilisez le câble AMSOFT DI2.

Le connecteur le plus large du câble doit s'insérer dans le connecteur DISK DRIVE 2 situé à l'arrière de l'ordinateur, tandis que le plus étroit vient s'insérer dans le connecteur situé à l'arrière de l'unité FD1.

N'OUBLIEZ PAS de retirer les disquettes des unités et de mettre le système hors tension avant de connecter ou déconnecter la deuxième unité de disquette. Si vous effectuez des connexions alors que le système est sous tension, vous risquez d'altérer le programme stocké dans la mémoire de l'ordinateur. Avant de vous attaquer aux connexions, pensez toujours à sauvegarder les programmes importants.

Après connexion de l'unité FD1 à l'ordinateur, mettez-la sous tension, à l'aide de son commutateur arrière. Mettez ensuite sous tension le 6128 à l'aide du commutateur situé à sa droite. Les voyants rouge et vert de l'unité FD1 doivent alors s'allumer, indiquant que l'unité est prête à fonctionner.

Le fonctionnement de la seconde unité de disquette est traité en détail dans une partie ultérieure du manuel.

Enceintes/amplificateur extérieur

Pour apprécier pleinement les capacités sonores à trois canaux de votre ordinateur, vous pouvez le raccorder à un amplificateur muni d'enceintes.

Le câble d'entrée de l'amplificateur doit se terminer par une fiche jack stéréo de 3,5 mm, que vous insérerez dans la prise STEREO de l'ordinateur.

Voici les connexions de la fiche jack :

Canal gauche - Extrémité de la fiche
Canal droit - Partie centrale de la fiche
Masse électrique - Partie arrière de la fiche

Le 6128 émet alors des signaux de niveau constant par la sortie STEREO ; c'est à vous qu'il appartient de régler le volume sonore, la balance, les aigus et les graves à l'aide des commandes de votre amplificateur.

Vous pouvez également utiliser un casque d'écoute à forte impédance, dont le volume ne sera toutefois pas réglable à partir de la commande VOLUME de l'ordinateur. Quant aux casques à faible impédance comme ceux qui équipent généralement les chaînes stéréo, ils ne fonctionnent pas directement sur l'ordinateur.

Une partie de ce manuel est d'ailleurs consacrée à l'acheminement du son sur les différents canaux de sortie.

Périphériques d'extension

Vous pouvez également connecter des périphériques d'extension tels qu'interfaces série, modems, crayons lumineux, cartes ROM, etc., sur le connecteur EXPANSION situé à l'arrière de l'ordinateur.

Cette prise recevra aussi le synthétiseur de parole/amplificateur SSA2 AMSOFT.

Le chapitre « Pour information... » détaille les connexions à la prise EXPANSION.

Vérifiez, pour finir, que vous avez bien suivi les avertissements figurant au début de ce manuel, dans la rubrique « IMPORTANT » :

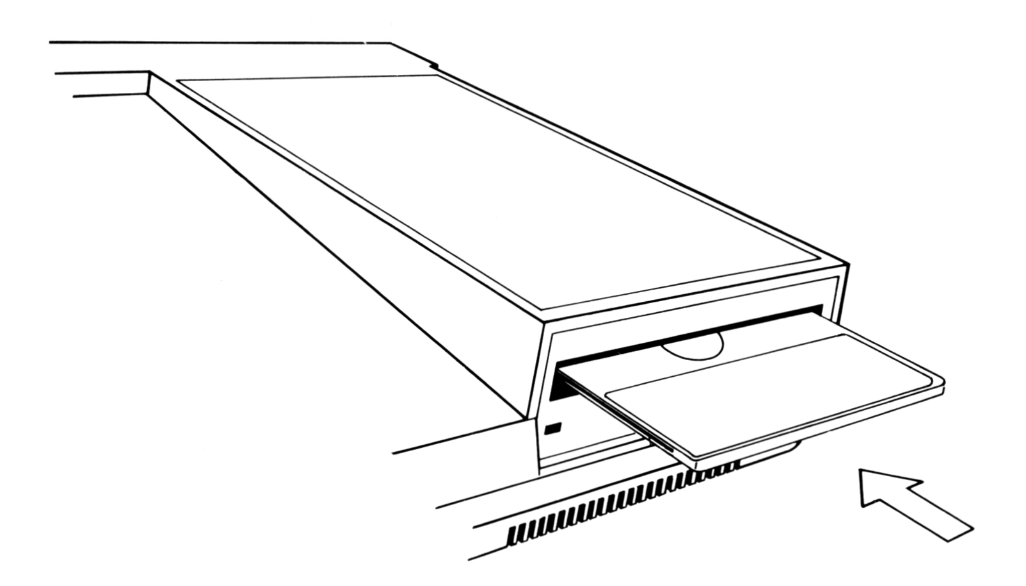
REMARQUES RELATIVES A L'INSTALLATION : 6 et 7
REMARQUES RELATIVES AU FONCTIONNEMENT : 4 et 8

Partie 3 : Les disquettes

Le 6128 utilise des disquettes compactes de format 3 pouces. Nous vous conseillons fortement, pour une question de fiabilité, de n'utiliser que des CF-2 (Compact Floppy) d'AMSOFT. Cependant, les disquettes 3 pouces des principaux fabricants peuvent également convenir.

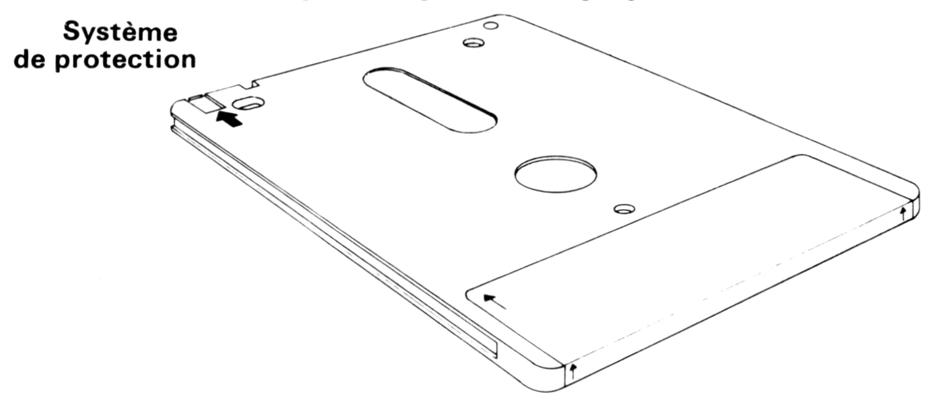
Insertion

Chaque face de la disquette peut être utilisée séparément. La disquette doit être insérée avec son étiquette vers l'extérieur et la face que vous voulez utiliser dirigée vers le haut.



Protection en écriture

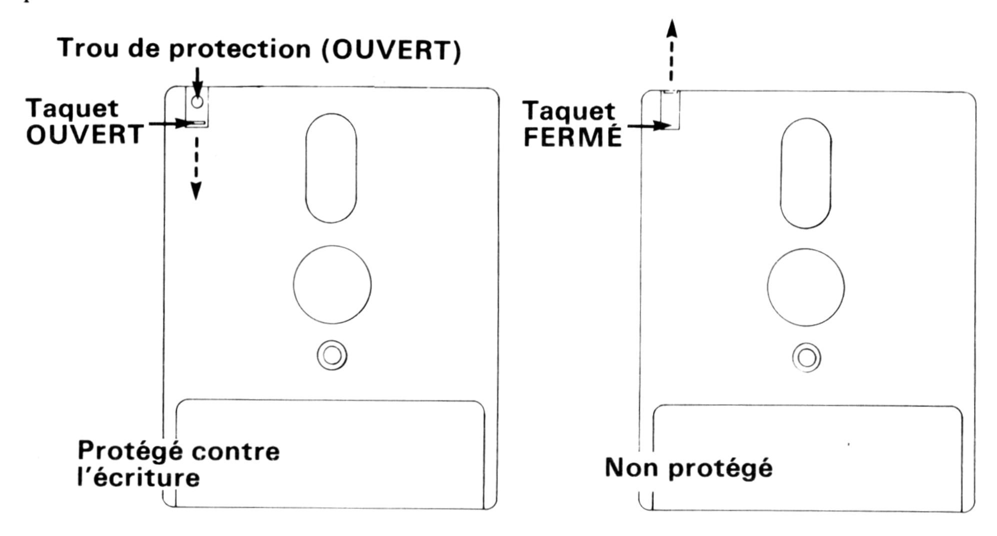
Le coin gauche de chaque face d'une disquette porte une flèche montrant un trou muni d'un obturateur. C'est le trou de protection en écriture qui vous empêche d'effacer ou d'écrire sur des données qu'il est important de ne pas perdre.



Quand le trou est fermé, l'ordinateur peut écrire sur la disquette. Quand le trou est ouvert, rien ne peut être écrit, vous protégeant contre tout effacement accidentel de programmes importants.

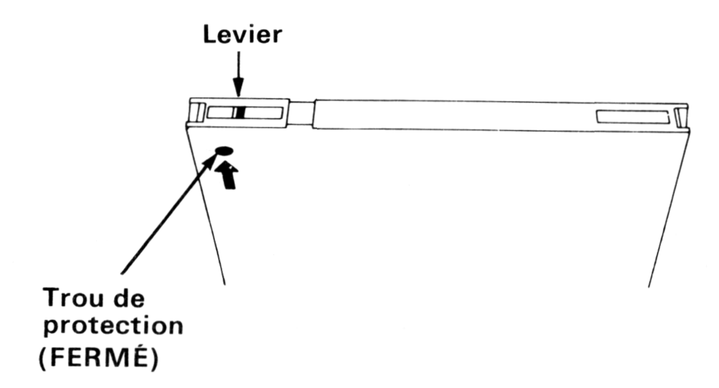
Les fabricants de disquettes emploient des procédés différents pour l'ouverture et la fermeture des trous de protection. Pour la CF-2 d'AMSOFT, suivez les instructions ci-dessous :

Pour ouvrir le trou de protection, abaissez l'obturateur situé sur le coin gauche de la disquette.

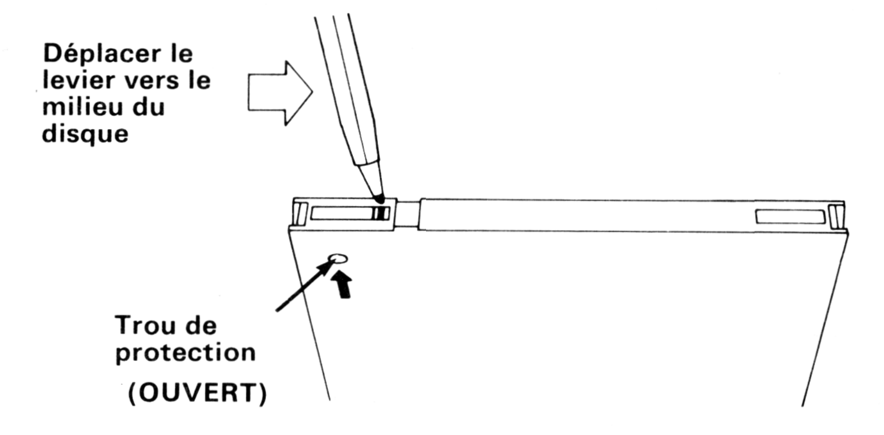


Pour fermer le trou de protection, relevez l'obturateur.

Certains autres modèles utilisent un petit levier en plastique situé dans une fente du coin gauche.



Pour ouvrir le trou de protection sur ce genre de disquette, il faut faire glisser le levier vers le milieu de la disquette en utilisant la pointe d'un stylo, par exemple.



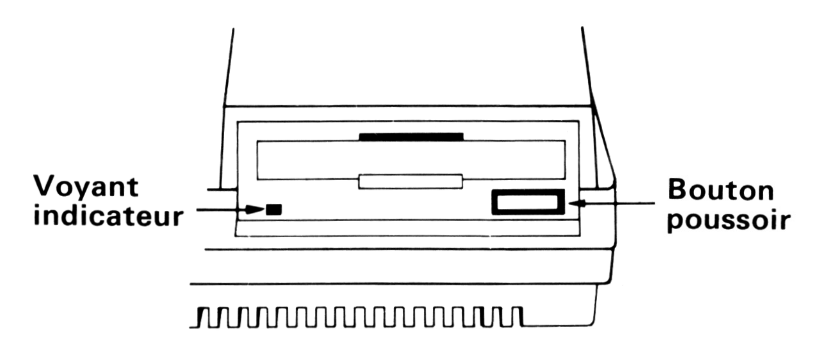
Quelle que soit la procédure employée, il faut donc ouvrir le trou de protection si vous voulez éviter de perdre vos programmes ou vos données en écrivant par-dessus.

IMPORTANT

Assurez-vous toujours que les trous de protection de votre disquette système CP/M sont ouverts.

Quand la disquette est à l'intérieur de l'unité de disquette

Sur le devant de l'unité de disquette se trouvent un voyant rouge et un bouton-poussoir pour l'éjection des disquettes.



Le voyant indicateur

Ce petit voyant rouge a deux fonctions : Si une seule unité de disquette est connectée, le voyant indique que des données sont lues ou écrites sur la disquette. Si deux unités de disquettes sont connectées, le voyant de l'unité **B** sera toujours allumé, vous permettant ainsi d'identifier les unités rapidement. Cependant, lors d'une opération (lecture ou écriture) de l'unité A, le voyant de l'unité **B** doit s'éteindre.

Le bouton-poussoir

Il vous permet d'éjecter la disquette de son unité.

IMPORTANT

1. TOUJOURS ENLEVER VOTRE DISQUETTE DE L'UNITE AVANT D'ALLUMER OU D'ETEINDRE VOTRE SYSTEME.

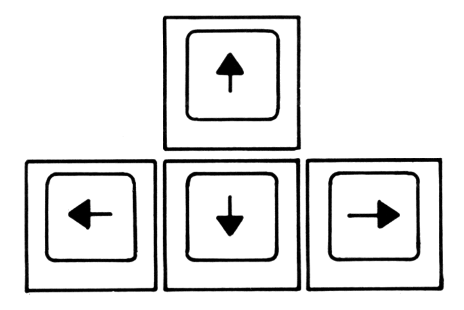
2. NE JAMAIS ESSAYER D'EJECTER UNE DISQUETTE PENDANT UNE OPERATION DE LECTURE OU D'ECRITURE SUR CETTE DISQUETTE, CE QUI POURRAIT ENTRAINER LA DESTRUCTION DES DONNEES S'Y TROUVANT.

3. LA DISQUETTE MAGNETIQUE EST PROTEGEE PAR UNE ENVELOPPE DE PLASTIQUE DUR. ATTENTION A NE PAS TOUCHER A LA DISQUETTE ELLE-MEME.

Partie 4 : Informations préalables

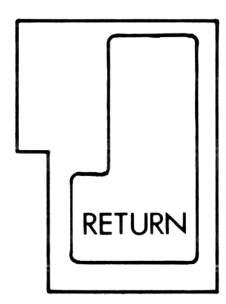
Avant de passer au chargement des logiciels et à la sauvegarde des programmes sur disquettes, attardons-nous sur le clavier. Ceux qui ont l'habitude des micro-ordinateurs peuvent sauter cette partie.

L'ordinateur est sous tension et le message initial affiché : voyons maintenant les fonctions des différentes touches du clavier.



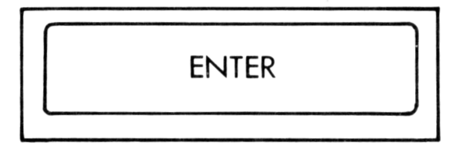
Les touches de déplacement du curseur ⇦ ⇨ ⇧ ⇩ , situées en haut et à droite du clavier, permettent de faire évoluer le curseur, petit bloc homogène, en différents points de l'écran.

Appuyez tour à tour sur chaque touche de déplacement et observez le déplacement du curseur.

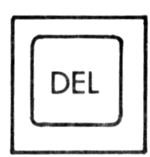


La touche **[RETURN]** permet d'entrer dans la mémoire de l'ordinateur les informations que vous venez de taper. Vous devez appuyer sur cette touche après chaque instruction tapée au clavier : aussitôt, une nouvelle ligne commence à l'écran.

A partir de maintenant, quand vous verrez **[RETURN]**, cela voudra dire appuyez sur la touche **[RETURN]** après chaque instruction ou ligne de programme.

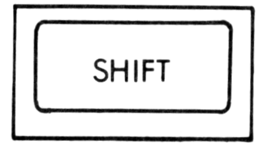


En conditions normales de fonctionnement (par défaut), cette touche a les mêmes effets que **[RETURN]** et peut se substituer à elle. Toutefois, à l'instar des touches de fonction du clavier numérique, les fonctions de la touche **[ENTER]** peuvent être redéfinies. Cette question sera abordée ultérieurement.



Cette touche est utilisée pour effacer un caractère à gauche du curseur (par exemple une lettre ou un nombre) que vous ne voulez pas conserver.

Tapez **abcd** ; la lettre **d** se trouve à gauche du curseur. Si vous décidez que vous ne voulez pas de cette lettre, appuyez sur la touche **[DEL]** une fois et la lettre **d** disparaît. Si vous continuez à appuyer sur cette touche **[DEL]**, les lettres **abc** disparaîtront aussi.



Il y a deux touches **[SHIFT]**. Si vous appuyez sur l'une des deux et la maintenez appuyée pendant que vous tapez un autre caractère, une majuscule ou le symbole supérieur de la touche s'affiche sur l'écran.

Tapez la lettre **e**, puis appuyez sur **[SHIFT]** (en maintenant cette touche) et tapez sur la lettre **e** de nouveau. Sur l'écran vous verrez :

```
eE
```

Tapez maintenant quelques espaces en maintenant la barre d'espacement appuyée. Essayez avec les touches du haut du clavier, au-dessus des lettres. Tapez le chiffre **2**, puis appuyez sur **[SHIFT]** et tapez le chiffre **2** de nouveau. Vous obtenez sur l'écran :

```
2"
```

Vous pouvez maintenant voir ce qui arrive quand vous appuyez sur la touche **[SHIFT]** tout en tapant un caractère. Essayez en tapant chacune des touches, d'abord toutes seules, puis en appuyant sur la touche **[SHIFT]** en même temps.



Cette touche a le même effet que la touche **[SHIFT]**, excepté qu'il suffit de la presser une fois. Toutes les lettres seront affichées en majuscules, sauf les touches numériques qui resteront des chiffres. Appuyez simultanément sur **[CONTROL]** et **[CAPS LOCK]** puis tapez :

```
abcdef123456
```

Sur l'écran vous voyez :

```
ABCDEF123456
```

Et les chiffres ont été tapés tels quels. Maintenant, si vous appuyez sur **[SHIFT]** en même temps, vous aurez :

```
abcdef123456
```

Pour revenir aux minuscules, appuyez de nouveau sur **[CAPS LOCK]**.





Pour taper les majuscules et les symboles supérieurs des touches sans avoir à presser continuellement sur **[SHIFT]**, appuyez sur **[CONTROL]**, puis sur **[CAPS LOCK]**. Maintenant tapez :

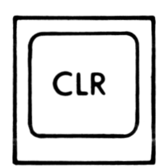
```
abcdef123456
```

Vous verrez apparaître :

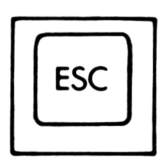
```
ABCDEF!"#$%&
```

Il est également possible de taper des chiffres sur le pavé numérique de droite du clavier (**f0** à **f9**).

Si vous maintenez la touche **[CONTROL]** enfoncée et appuyez une fois sur **[CAPS LOCK]**, vous revenez à l'état précédent, majuscules ou minuscules. Si vous êtes toujours en majuscules, appuyez sur **[CAPS LOCK]** une fois pour revenir en minuscules.



Cette touche sert à effacer un caractère qui se trouve sous le curseur. Tapez : **ABCDEFGH**. Le curseur est à droite du (**H**). Puis appuyez quatre fois sur le curseur gauche ⇦. Le curseur se superpose à la lettre **E**. Notez que la lettre **E** est toujours visible sous le curseur. Appuyez sur la touche **[CLR]** une fois et vous verrez que la lettre **E** a été effacée et que les lettres **FGH** se sont déplacées d'une case vers la gauche tandis que la lettre **F** se trouve sous le curseur. Appuyez sur la touche **[CLR]** sans la lâcher. Les lettres **F**, puis **G** et **H** disparaissent.



Cette touche bleue en haut du clavier, à gauche, permet de prendre la poudre d'**ESC**ampette, autrement dit d'interrompre momentanément un programme en pressant la touche une fois et de le reprendre en tapant une autre touche, ou d'arrêter complètement en appuyant encore une fois sur **[ESC]**.

Appuyez maintenant deux fois de suite sur la touche **[ESC]**. L'ordinateur est à nouveau prêt à accepter de nouvelles instructions.

IMPORTANT

Quand vous arrivez au bord droit de l'écran en tapant plus de 40 caractères, le caractère suivant passe automatiquement à la ligne suivante. Cela veut dire qu'il NE FAUT PAS presser **[RETURN]**, à l'instar des habitués des machines à écrire, qui tapent RETOUR ou ENVOI à la fin d'une ligne.

L'ordinateur s'en charge automatiquement et réagit à un **[RETURN]** non prévu par un message d'erreur - (d'habitude **Syntax error**) - soit immédiatement, soit quand vous lancez l'exécution du programme.

Syntax Error (ou erreur de syntaxe)

Si le message **Syntax error** apparaît sur l'écran, l'ordinateur vous dit qu'il ne comprend pas une des instructions que vous avez tapées. Par exemple, si vous tapez :

```
printt [RETURN]
```

...le message suivant apparaît :

```
Syntax error
```

Car l'ordinateur ne connaît pas la commande **printt**.

Si vous vous trompez dans une ligne de programme comme :

```
10 printt "abc" [RETURN]
```

...le message **Syntax error** n'apparaîtra que lors de l'exécution du programme.

Tapez :

```
run [RETURN]
```

Cette commande ordonne l'exécution du programme que vous venez d'entrer en mémoire.

Sur l'écran vous voyez :

```
Syntax error in 10
10 printt "abc"
```

Ce message vous dit dans quelle ligne se trouve l'erreur et vous présente la ligne du programme avec le curseur-éditeur afin que que vous puissiez corriger l'erreur.

Appuyez sur le curseur droit ⇨ jusqu'à ce que le curseur soit au-dessus du **t** de **printt** ; pressez la touche **[CLR]** pour enlever le **t** superflu, puis pressez **[RETURN]** pour introduire la ligne correcte dans l'ordinateur.

Tapez :

```
run [RETURN]
```

...et vous verrez que l'ordinateur a accepté l'instruction en affichant :

```
abc
```

Pour terminer, vérifiez que vous avez bien respecté les avertissements figurant au début du manuel, dans la rubrique « IMPORTANT » :

REMARQUES RELATIVES A L'INSTALLATION : 4 et 5
REMARQUES RELATIVES AU FONCTIONNEMENT : 1

Partie 5 : Chargement des logiciels et des jeux...

Bienvenue aux utilisateurs qui abordent directement cette partie !

Pour expérimenter la vitesse de chargement des logiciels sur disquette, mettez le système sous tension et insérez l'une des disquettes CP/M dans l'unité, face 4 vers le haut.

Tapez :

```
run "rointime.dem" [RETURN]
```

Au bout de quelques secondes, le programme est chargé en mémoire. Répondez à la question affichée à l'écran, confirmant que vous utilisez bien un moniteur vert (tapez **Y** pour Oui et **N** pour Non). Vous verrez alors apparaître une démonstration du jeu « Roland in Time ». Elle vous incitera peut-être à acheter une copie du jeu...

Après cette démonstration, vous pouvez sortir du programme en appuyant sur la touche **[ESC]** tout en enfonçant les touches **[CONTROL]** et **[SHIFT]**. Vous réinitialisez ainsi votre système afin de lancer une nouvelle application sans avoir besoin d'enlever la disquette de son unité.

Si le programme n'est pas chargé, un message d'erreur vous indique l'anomalie :

```
Drive A: Disc missing
Retry, Ignore or Cancel?
```

...signifie que vous n'avez pas inséré la disquette correctement ou, si vous disposez d'une deuxième unité de disquette, que vous l'avez insérée dans l'unité B.

```
ROINTIME.DEM not found
```

...indique que vous n'avez pas inséré la bonne disquette (ou la bonne face) ou que vous avez fait une erreur en tapant le nom, **ROINTIME.DEM**.

```
Bad command
```

...signifie que vous avez mal tapé **ROINTIME.DEM** en insérant un espace ou un signe de ponctuation superflu.

```
Type mismatch
```

...signifie que vous avez oublié les guillemets.

```
Syntax error
```

...vous avez fait une erreur en tapant run.

```
Drive A: read fail
Retry, Ignore or Cancel?
```

...signifie que l'ordinateur n'a pas réussi à lire les données de la disquette. Vérifiez que vous avez inséré la bonne disquette et tapez **R** pour Retry (nouvel essai). Ce message apparaît si vous avez endommagé la disquette en mettant le système sous ou hors tension sans la retirer.

Lorsque vous saurez dupliquer une disquette, n'oubliez jamais de copier les programmes importants, notamment le CP/M.

Chargement du logiciel AMSOFT et du programme « BIENVENUE »

Nous espérons avoir suscité votre intérêt pour l'informatique, passons maintenant à un jeu...

Insérez votre disquette dans l'unité et tapez :

```
run "disc" [RETURN]
```

Au bout de quelques secondes, le jeu est chargé et vous pouvez commencer.

Tapez : **run " disc "** après avoir inséré l'une des disquettes CP/M dans l'unité, face 4 vers le haut, ce qui déclenche une démonstration de « Bienvenue ».

Après cette démonstration, réinitialisez votre système à l'aide des touches **[CONTROL]**, **[SHIFT]** et **[ESC]**.

Bien que l'instruction **run " disc "** permette de charger la plupart des logiciels AMSOFT sur disquettes, il peut arriver que vous ayez à en utiliser une autre. Dans tous les cas, respectez scrupuleusement les instructions de chargement indiquées sur la disquette du logiciel.

Pour terminer, vérifiez que vous avez bien suivi les avertissements figurant au paragraphe « IMPORTANT » du début de ce manuel.

REMARQUES RELATIVES A L'INSTALLATION : 6
REMARQUES RELATIVES AU FONCTIONNEMENT : 1, 5, 6

Partie 6 : Passons à la programmation...

Jusqu'ici, nous avons vu ce qu'il faut faire et ne pas faire, comment installer l'ordinateur et connecter les périphériques. Nous avons appris à charger les logiciels et à utiliser certaines touches du clavier. Nous allons maintenant apprendre à utiliser des instructions pour faire bouger les choses...

Comme vous et moi, l'ordinateur ne comprend les instructions que dans la langue qu'il connaît. Ce langage s'appelle le BASIC (abréviation de Beginners' All-purpose Symbolic Instruction Code). Les mots du BASIC sont des « mots clés » communiquant chacun une fonction particulière à l'ordinateur. Comme tous les langages, le BASIC obéit à des règles de grammaire. Dans ce cas précis, la grammaire s'appelle « Syntaxe » et l'ordinateur a toujours l'amabilité de vous indiquer vos **erreurs de syntaxe**.

Introduction aux mots clés du BASIC Schneider

Vous trouverez une description de la syntaxe du BASIC Schneider dans le chapitre « Liste complète des mots clés du BASIC Schneider 6128 ».

CLS

Pour effacer l'écran, tapez :

```
cls [RETURN]
```

L'écran s'efface et le mot **Ready** s'affiche suivi du curseur ■, dans l'angle supérieur gauche de l'écran.

Vous pouvez utiliser indifféremment les MAJUSCULES ou les minuscules pour entrer un mot clé BASIC dans votre ordinateur.

PRINT

Sert chaque fois qu'il est nécessaire d'afficher des caractères, des mots ou des chiffres sur l'écran. Si vous tapez l'instruction suivante :

```
print "bonjour" [RETURN]
```

Vous voyez apparaître :

```
bonjour
```

Les guillemets (" ") délimitent le message que l'on désire voir apparaître. Vous remarquerez que bonjour est apparu sur l'écran aussitôt que la touche **[RETURN]** a été frappée. Tapez :

```
cls [RETURN]
```

...pour effacer l'écran.

RUN

L'exemple précédent montrait une instruction simple d'une seule ligne. L'instruction a été exécutée dès que vous avez appuyé sur la touche **[RETURN]**, puis a été oubliée. Il est possible de stocker une série d'instructions s'exécutant selon un ordre spécifique, que l'on appelle un « programme ». Les instructions qui composent un programme sont de même type que celles que nous venons de voir, mais précédées d'un numéro de ligne. Les numéros de ligne d'un programme indiquent à l'ordinateur l'ordre d'exécution des instructions. Lorsque vous appuyez sur **[RETURN]**, la ligne est stockée dans la mémoire. Tapez :

```
10 print "bonjour" [RETURN]
```

Vous remarquerez qu'après avoir appuyé sur **[RETURN]**, bonjour ne s'est pas affiché à l'écran. Il s'est inscrit dans la mémoire de l'ordinateur pour former un programme d'une ligne. Pour exécuter ce programme, tapez :

```
run [RETURN]
```

Vous verrez alors **bonjour** apparaître à l'écran. Vous pouvez, si vous le désirez, remplacer **PRINT** par un point d'interrogation :

```
10 ? "bonjour" [RETURN]
```

LIST

Lorsqu'un programme a été mis en mémoire, on peut vérifier ce qu'on a tapé en demandant sa « liste ». Tapez :

```
list [RETURN]
```

...et vous voyez apparaître :

```
10 PRINT "bonjour"
```

...qui est le programme stocké en mémoire.

Avez-vous remarqué que le mot **PRINT** est maintenant en majuscules ? Cela signifie qu'il a été reconnu par l'ordinateur en tant que mot clé du BASIC.

Tapez : **cls [RETURN]** pour effacer l'écran, sans supprimer votre programme de la mémoire.

GOTO

Le mot clé GOTO demande à l'ordinateur d'aller d'une ligne à une autre afin d'en sauter un certain nombre ou de former une boucle. Tapez :

```
10 PRINT"bonjour" [RETURN]
20 GOTO 10 [RETURN]
```

...puis :

```
run [RETURN]
```

Vous voyez alors bonjour s'afficher sans interruption, ligne après ligne, la ligne **20** du programme demandant à l'ordinateur de retourner à la ligne **10** et de poursuivre l'exécution.

Pour interrompre momentanément le programme, appuyez sur **[ESC]**. Pour continuer, actionnez n'importe quelle autre touche. Pour l'arrêter complètement, actionnez **[ESC]** deux fois de suite.

Faites :

```
cls [RETURN]
```

...pour effacer l'écran.

Pour voir le mot **bonjour** s'afficher sur toute la ligne, il suffit de mettre un point-virgule à la fin de la ligne 10, après les guillemets.

Tapez :

```
10 PRINT"bonjour"; [RETURN]
20 GOTO 10 [RETURN]
run [RETURN]
```

Le point-virgule commande à l'ordinateur d'afficher le prochain groupe de caractères immédiatement après le précédent (à moins qu'il ne soit trop long pour tenir sur la ligne).

Quittez ce programme en appuyant deux fois sur **[ESC]**. Tapez la ligne **1Ø** à nouveau, mais en remplaçant le point-virgule (;) par une virgule (,).

```
10 PRINT"bonjour", [RETURN]
run [RETURN]
```

La virgule a demandé à l'ordinateur d'afficher le prochain groupe de caractères 13 colonnes après le début de celui-ci. Cette fonction permet d'afficher les données en colonnes. Cependant, si le nombre de caractères inclus dans un groupe dépasse 12, le groupe suivant est décalé de 13 autres colonnes, afin de toujours ménager un espace entre les groupes de caractères.

La taille de ces zones de 13 caractères peut être modifiée par la commande **ZONE**, décrite plus loin dans le manuel.

Pour sortir du programme, appuyez deux fois sur **[ESC]**. Pour vider complètement la mémoire, remettez l'ordinateur à zéro en appuyant sur **[CONTROL]**, **[SHIFT]** et **[ESC]**, dans cet ordre.

INPUT

Cette commande sert à informer l'ordinateur qu'il doit attendre que l'on ait tapé quelque chose avant de continuer. Par exemple :

```
10 INPUT "Quel age avez-vous";age [RETURN]
20 PRINT"Vous paraissez nettement moins que vos";age;"ans."
[RETURN]
run [RETURN]
```

Sur l'écran on voit :

```
Quel age avez-vous?
```

Si vous donnez votre âge, mettons 18, puis **[RETURN]**, on voit alors s'afficher :

```
Vous parraissez nettement moins que vos 18 ans.
```

Cet exemple montre l'utilisation de la fonction **input** et d'une variable numérique. Le mot âge est mis en mémoire à la fin de la ligne **1Ø** pour que l'ordinateur l'associe à tout nombre tapé après le point d'interrogation, afin de procéder à l'affichage de la ligne **2Ø**. Bien que nous ayons utilisé le mot age pour la variable **âge**, nous aurions pu aussi bien prendre la lettre **a** ou **b**...

Réinitialisez l'ordinateur avec les touches **[CONTROL]** **[SHIFT]** et **[ESC]**. Si vous vouliez l'entrée (INPUT) de caractères indéterminés (lettres ou lettres et chiffres), le signe dollar (**$**) doit être placé à la fin de la variable. Cette variable est alors appelée « variable chaîne ».

Tapez le programme suivant (en faisant attention à mettre un espace après le **r** de **bonjour** et avant le **m** de **mon**).

```
10 INPUT "Quel est ton nom";nom$ [RETURN]
20 PRINT"bonjour ";nom$;" mon nom est Roland" [RETURN]
run [RETURN]
```

Sur l'écran, on voit :

```
Quel est ton nom?
```

Tapez votre nom puis **[RETURN]**

Si votre nom est Fred, vous verrez sur l'écran :

```
bonjour Fred mon nom est Roland
```

Bien que nous ayons utilisé **nom$** comme variable chaîne, nous aurions aussi bien pu utiliser **a$**. Nous allons maintenant combiner les deux exemples précédents en un seul programme.

Faisons à nouveau **[CONTROL]** **[SHIFT]** et **[ESC]**. Puis tapons le programme suivant :

```
5 CLS [RETURN]
10 INPUT "Quel est ton nom";a$ [RETURN]
20 INPUT "Quel est ton age";b [RETURN]
30 PRINT"Je dois dire ";a$;" que tu ne fais pas tes";b;"ans.
" [RETURN]
run [RETURN]
```

Dans ce programme, nous avons utilisé deux variables, **a$** pour le nom et **b** pour l'âge. Sur l'écran on voit :

```
Quel est ton nom?
```

Tapez votre nom (**Fred**) puis **[RETURN]**. La question suivante apparaît :

```
Quel est ton age?
```

Tapez maintenat votre âge (**18**) puis **[RETURN]**.

```
Je dois dire Fred que tu ne fais pas tes 18 ans.
```

Editer un programme

Si l'une des lignes du programme a été tapée incorrectement, entraînant le message **Syntax error** ou un autre message d'erreur, il est possible de l'éditer (autrement dit de la modifier) sans avoir à la retaper. Supposons que le programme précédent ait été mal tapé :

```
5 CLSS [RETURN]
10 INPUT "Quel est to nom";a$ [RETURN]
20 INPUT "Quel est ton age";b [RETURN]
30 PRINT"Je dois dire";a$;" que tu ne fais pas tes";b;"ans."
 [RETURN]
```

Trois erreurs se sont glissées dans le programme ci-dessus :

Dans la ligne **5**, on a tapé **clss** au lieu de **cls**
Dans la ligne **10**, on a tapé **to** au lieu de **ton**
Dans la ligne **30**, on a oublié l'espacement entre **dire** et les guillemets (**"**)

Il existe trois méthodes pour éditer un programme. La première consiste à retaper entièrement la ligne. Quand une ligne est retapée et entrée en mémoire, elle remplace la ligne qui portait le même numéro.

La deuxième méthode consiste à éditer à l'aide du curseur.

La dernière est appelée Copy Cursor, autrement dit Copie avec l'aide du Curseur.

Méthode d'édition à l'aide du curseur

Pour corriger la ligne **5**, tapez :

```
edit 5 [RETURN]
```

La ligne **5** apparaît sous la ligne **30**, le curseur placé sur le **c** de **clss**.

Pour enlever le **s** en trop dans **clss**, appuyez sur la touche curseur droite jusqu'à ce que celui-ci soit sur le dernier **s**, puis appuyez sur la touche **[CLR]**. Le **s** a disparu.

Appuyez maintenant sur **[RETURN]**, afin de corriger la ligne **5** dans la mémoire. Tapez maintenant :

```
list [RETURN]
```

...pour vérifier que la ligne **5** est correcte.

La commande **AUTO**, décrite plus loin, peut servir à éditer un nombre de lignes successives de la même manière que la méthode d'édition à l'aide du curseur.

Méthode par copie avec le curseur

Le curseur de copie est un deuxième curseur qui s'affiche lorsque vous appuyez simultanément sur **[SHIFT]** et sur l'une des touches du curseur. Il se détache du curseur principal et peut circuler sur l'écran de manière autonome.

Pour corriger les fautes des lignes **10** et **30**, actionnez et maintenez la touche **[SHIFT]** en pressant sur la touche curseur ⇧, jusqu'à ce que le curseur soit au début de la ligne **10**. Vous voyez que le curseur principal, en bas, n'a pas bougé. Puis pressez la touche **[COPY]** jusqu'à ce que le curseur soit placé dans l'espace entre to et nom. La ligne **10** est réécrite en même temps en bas et le curseur principal s'immobilise à la même place que le curseur de copie. Tapez la lettre **n**, qui apparaît sur la ligne du bas seulement.

Le curseur principal a bougé mais le curseur de copie est resté à sa place. Maintenant, appuyez sur la touche **[COPY]** pour afficher la totalité de la ligne **10**. Pressez **[RETURN]** afin que cette nouvelle ligne **10** soit mise en mémoire. Le curseur de copie disparaît et le curseur principal se place au-dessous de la ligne **10**. Pour corriger la deuxième faute, maintenez la touche **[SHIFT]** enfoncée et appuyez sur la touche curseur ⇧ pour faire apparaître le curseur de copie au début de la ligne **30**.

Appuyez sur **[COPY]** jusqu'à ce que le curseur de copie soit superposé aux guillemets suivant « dire ». Appuyez une seule fois sur la barre d'espacement. Un espace va s'introduire sur la ligne du bas. Continuez d'appuyer sur la touche **[COPY]** jusqu'à ce que la ligne **30** soit copiée. Appuyez sur **[RETURN]**

Vous pouvez demander la liste du programme en tapant :

```
list [RETURN]
```

REMARQUE : Pour déplacer rapidement le curseur (en cours d'édition) à l'extrémité droite ou gauche d'une ligne, maintenez la touche **[CONTROL]** et appuyez sur une des touches fléchées ⇦ ou ⇨.

Effectuez la remise à zéro avec les touches **[CONTROL]**, **[SHIFT]** et **[ESC]**.

IF

Les commandes **IF** et **THEN** demandent à l'ordinateur de tester une condition spécifiée et de procéder à une exécution en fonction du résultat du test. Par exemple, dans l'instruction :

```
if 1+1=2 then print "vrai" [RETURN]
```

...l'ordinateur vérifie la condition 1 + 1 = 2 et affiche « vrai », le cas échéant.

Le mot clé **ELSE** peut servir à indiquer ce que l'ordinateur doit faire si la condition **IF** n'est pas remplie.

```
if 1+1=0 then print "vrai" else print "faux" [RETURN]
```

Nous allons maintenant prolonger notre programme à l'aide de la commande **IF THEN**.

Tapez le programme suivant, en remarquant que nous introduisons deux symboles nouveaux : < signifie moins grand que et se situe à côté de la lettre M, > signifie plus grand que et se trouve à côté du signe <, moins grand que.

```
5 CLS [RETURN]
10 INPUT "quel est ton nom";a$ [RETURN]
20 INPUT "quel age as-tu";age [RETURN]
30 IF age < 13 THEN 60 [RETURN]
40 IF age < 20 THEN 70 [RETURN]
50 IF age > 20 THEN 80 [RETURN]
60 PRINT"donc ";a$;" tu n'est pas encore un adolescent avec
tes";age;"ans.":END [RETURN]
70 PRINT"donc ";a$;" tu es un adolescent avec tes";age;"ans.
":END [RETURN]
80 PRINT"eh bien ";a$;"tu n'est donc plus  un  adolescent  a
vec  tes";age;"ans."
[RETURN]
```

Pour vérifier que ce programme est correct, faites :

```
list [RETURN]
```

...puis tapez :

```
run [RETURN]
```

Vous pouvez maintenant répondre aux questions de l'ordinateur et observer ce qui se passe.

Vous pouvez constater les effets du **IF** (si) et du **THEN** (alors) comme commandes dans un programme. Nous avions aussi ajouté le mot **END** (fin) à la fin des lignes **60** et **70**. Ce mot réservé **END** est utilisé pour mettre fin à un programme. S'il n'était pas là, le programme continuerait à avancer et afficherait aussi les lignes **70** et **80**.

Le mot **END** à la fin de la ligne **70** joue le même rôle. Les deux-points (:) avant le mot **END** le séparent de l'instruction précédente. Les : sont utilisés pour séparer des instructions afin d'en mettre plusieurs sur une même ligne. Nous avons aussi ajouté la ligne **5** pour effacer l'écran, comme il convient de le faire au début de chaque programme, pour rendre les choses plus claires.

Remise à zéro avec **[CONTROL]**, **[SHIFT]** et **[ESC]**.

FOR et NEXT

Nous allons utiliser maintenant les commandes **FOR** et **NEXT**.

Les commandes **FOR** et **NEXT** permettent de répéter un certain nombre de fois une action définie (boucle). Le groupe d'instructions à répéter doit être délimité par **FOR** et **NEXT**.

Tapez :

```
5 CLS [RETURN]
10 FOR a=1 TO 10 [RETURN]
20 PRINT"ACTION executee";a;"fois" [RETURN]
30 NEXT a [RETURN]
run [RETURN]
```

Vous remarquerez que l'instruction de la ligne **20** a été exécutée dix fois, comme le spécifie la commande **FOR** de la ligne **10**. Vous constatez également que la valeur de la variable est incrémentée de 1 à chaque passage dans la boucle.

Le mot clé **STEP** peut servir à définir le pas de la commande **FOR NEXT**. Par exemple, vous pouvez remplacer la ligne **10** par :

```
10 FOR a=10 TO 50 STEP 5 [RETURN]
run [RETURN]
```

Vous pouvez aussi fixer des pas négatifs :

```
10 FOR a=100 TO 0 STEP -10 [RETURN]
run [RETURN]
```

REM

REM est l'abréviation de **REM**ark. Cette instruction demande à l'ordinateur d'ignorer tout ce qui suit sur la ligne d'instruction. **REM** peut ainsi servir à introduire des informations : le titre d'un programme ou l'utilisation d'une variable par exemple :

```
10 REM Pan sur les envahisseurs [RETURN]
20 V=5:REM nombre de vies [RETURN]
```

L'apostrophe (obtenue en appuyant sur **[SHIFT]** et sur la touche **7**) peut remplacer **REM** :

```
10 ' Pan sur les envahisseurs [RETURN]
20 V=5' nombre de vies [RETURN]
```

GOSUB

Si un ensemble d'instructions doit être exécuté plusieurs fois, vous pouvez éviter de retaper les instructions en créant un « sous-programme » dont l'appel se fait grâce à la commande **GOSUB**, suivie du numéro de ligne requis. La fin du sous-programme s'indique par **RETURN**. L'ordinateur passe alors à l'instruction qui suit immédiatement la commande **GOSUB** qu'il vient d'exécuter.

(Les deux programmes suivants ne « font » rien d'autre qu'afficher des mots à l'écran. Comme ils ne servent qu'à illustrer le fonctionnement des sous-programmes, il n'est pas indispensable de les taper).

Par exemple, dans ce programme :

```
10 MODE 2 [RETURN]
20 PRINT"Dame souris trotte," [RETURN]
30 PRINT"Noire dans le gris du soir," [RETURN]
40 PRINT"Dame souris trotte," [RETURN]
50 PRINT"Grise dans le noir." [RETURN]
60 PRINT [RETURN]
70 PRINT"Un nuage passe," [RETURN]
80 PRINT"Il fait noir comme en un four" [RETURN]
90 PRINT"Un nuage passe," [RETURN]
100 PRINT"Tiens,le petit jour !" [RETURN]
110 PRINT [RETURN]
120 PRINT"Dame souris trotte," [RETURN]
130 PRINT"Rose dans les rayons bleus," [RETURN]
140 PRINT"Dame souris trotte," [RETURN]
150 PRINT"Debout paresseux !" [RETURN]
160 PRINT [RETURN]
170 PRINT"          P.Verlaine" [RETURN]
run [RETURN]
```

...vous pouvez constater qu'un certain nombre de lignes reviennent plusieurs fois, comme le vers de la ligne **190** et celui de la ligne **210**. Créons deux sous-programmes pour les vers, sans oublier **RETURN** à la fin. Ainsi, nous pouvons appeler les sous-programmes à l'aide de la commande **GOSUB 190** et **GOSUB 210** chaque fois que nous voulons utiliser le vers. Le programme prend alors cet aspect :

```
10 MODE 2 [RETURN]
20 GOSUB 190 [RETURN]
30 PRINT"Noire dans le gris du soir," [RETURN]
40 GOSUB 190 [RETURN]
50 PRINT"Grise dans le noir." [RETURN]
60 PRINT [RETURN]
70 GOSUB 210 [RETURN]
80 PRINT"Il fait noir comme en un four" [RETURN]
90 GOSUB 210 [RETURN]
100 PRINT"Tiens,le petit jour !" [RETURN]
110 PRINT [RETURN]
120 GOSUB 190 [RETURN]
130 PRINT"Rose dans les rayons bleus," [RETURN]
140 GOSUB 190 [RETURN]
150 PRINT"Debout paresseux !" [RETURN]
160 PRINT [RETURN]
170 PRINT"          P.Verlaine" [RETURN]
180 END [RETURN]
190 PRINT"Dame souris trotte," [RETURN]
200 RETURN [RETURN]
210 PRINT"Un nuage passe," [RETURN]
220 RETURN [RETURN]
run [RETURN]
```

Nous avons ainsi gagné un temps précieux ! Les sous-programmes bien pensés sont une partie essentielle de la programmation. Ils débouchent sur des programmes « structurés » et sont un bon réflexe de programmation.

Lorsque vous créez des sous-programmes, rappelez-vous qu'il vous est possible d'avoir plusieurs points d'entrée. Par exemple, un sous-programme occupant les lignes **500** à **800** pourra très bien être abordé au point **500** (**GOSUB500**), **640** (**GOSUB640**) ou **790** (**GOSUB790**).

Notez l'importance de l'instruction **END** à la ligne **180**. Sans elle, le programme continuerait après la ligne **170** et exécuterait la ligne **190**, qui ne doit l'être que par l'intermédiaire d'un **GOSUB**.

Arithmétique simple

Votre ordinateur peut facilement vous servir de calculatrice. Pour en assimiler le fonctionnement, nous vous proposons les exercices suivants. La réponse sera affichée aussitôt que la touche **[RETURN]** sera frappée.

Addition

(utilisez **[SHIFT]** et ;)

Tapez :

```
?3+3 [RETURN]
 6
```

(Vous n'avez PAS à taper le signe =)
Tapez :

```
?8+4 [RETURN]
 12
```

Soustraction

(Utilisez = pour moins)

Tapez :

```
?4-3 [RETURN]
 1
```

Tapez :

```
?8-4 [RETURN]
 4
```

Multiplication

(Utilisez **[SHIFT]** et : pour multiplier (* signifie x))

Tapez :

```
?3*3 [RETURN]
 9
```

Tapez :

```
?8*4 [RETURN]
 32
```

Division

(Utilisez / pour diviser sans appuyer sur **[SHIFT]**).

Tapez :

```
?3/3 [RETURN]
 1
```

Tapez :

```
?8/4 [RETURN]
 2
```

Division entière

(Utilisez la barre \ pour diviser avec suppression du reste)

Tapez :

```
?10\6 [RETURN]
 1
```

Tapez :

```
?20\3 [RETURN]
 6
```

Modulo

(Utilisez **MOD** pour obtenir le reste d'une division entière)

Tapez :

```
?10 MOD 4 [RETURN]
 2
```

Tapez :

```
?9 MOD 3 [RETURN]
 0
```

Racine carrée

Pour trouver la racine carrée d'un nombre, utilisez **sqr** ( ). Le nombre dont on veut extraire la racine doit être entre parenthèses.

Tapez :

```
?sqr(16) [RETURN](ceci veut dire √16)
 4
```

Tapez :

```
?sqr (100) [RETURN]
 10
```

Puissances

(Utilisez le signe ↑)

Exemples de puissances : 3↑2 (3 au carré), 3↑3 (3 au cube), etc.

Tapez :

```
?3↑3 [RETURN] (ceci veut dire 3³)
 27
```

Tapez :

```
?8↑4 [RETURN] (ceci veut dire 8⁴)
 4096
```

Racines cubiques

Vous pouvez facilement calculer les racines cubiques en adoptant la méthode ci-dessus :

Pour trouver la racine cubique de 27 ($\sqrt[3]{27}$)

Tapez :

```
?27↑(1/3) [RETURN]
 3
```

Pour trouver la racine cubique de 125

Tapez :

```
?125↑(1/3) [RETURN]
 5
```

Calculs composés

(+, -, *,/)

Les calculs mélangeant addition, multiplication, etc., sont compris par l'ordinateur, mais il faut faire attention à la priorité des opérateurs.

La priorité va à la multiplication, suivie de la division, de l'addition et enfin de la soustraction. Il existe cependant d'autres opérateurs dotés de priorités différentes.

Soit le calcul :

```
3+7-2*7/4
```

On pourrait penser qu'il se fait de la manière suivante :

```
3+7-2 *7 /4
= 8 *7 /4
= 56/4
= 14
```

La réalité est pourtant :

```
3+7-2*7/4
=3+7-14/4
=3+7-3.5
=10-3.5
=6.5
```

Pour le constater, il suffit de le taper :

```
?3+7-2*7/4 [RETURN]
 6.5
```

On peut changer la manière dont l'ordinateur calcule en ajoutant des parenthèses, affectant la priorité aux opérateurs qu'elles contiennent. Par exemple :

```
?(3+7-2)*7/4 [RETURN]
 14
```

Ordre de priorité de tous les opérateurs mathématiques :

↑	Elévation à la puissance
MOD	Modulo
-	Moins unaire (donne le négatif d'un nombre)
* et /	Multiplication et division
\	Division entière
+ et -	Addition et soustraction

Encore des puissances...

Quand on veut se servir de très grands ou très petits nombres dans les calculs, il est quelquefois utile de faire appel à la notation scientifique. La lettre **E** est utilisée en informatique pour les exposants des nombres en base 10. Vous êtes libre de taper **E** ou **e**, sans oublier que pour certains mathématiciens, **e** peut avoir un sens différent dans les logarithmes népériens.

Ainsi, 300 est égal à 3 x 10^2. En notation scientifique, on écrit **3E2**. De même, 0.03 est égal à 3 x 10^{-2}. On écrit **3E-2**. Essayez les exemples suivants.

Tapez :

```
?30*10 [RETURN]
 300
```

ou bien :

```
?3E1*1E1 [RETURN]
 300

?3000*1000 [RETURN]...ou... ?3E3*1E3 [RETURN]
 3000000

?3000*0.001 [RETURN]...ou... ?3E3*1E-3 [RETURN]
 3
```

Partie 7 : Sauvegarde

Maintenant que vous savez taper quelques instructions, vous avez sûrement très envie de sauvegarder vos programmes sur disquettes et les charger dans l'ordinateur à partir de ces mêmes disquettes.

Même si vous savez déjà sauvegarder et charger des programmes sur cassette, vous devez assimiler quelques notions supplémentaires pour effectuer les mêmes opérations sur disquette. Il existe essentiellement deux différences.

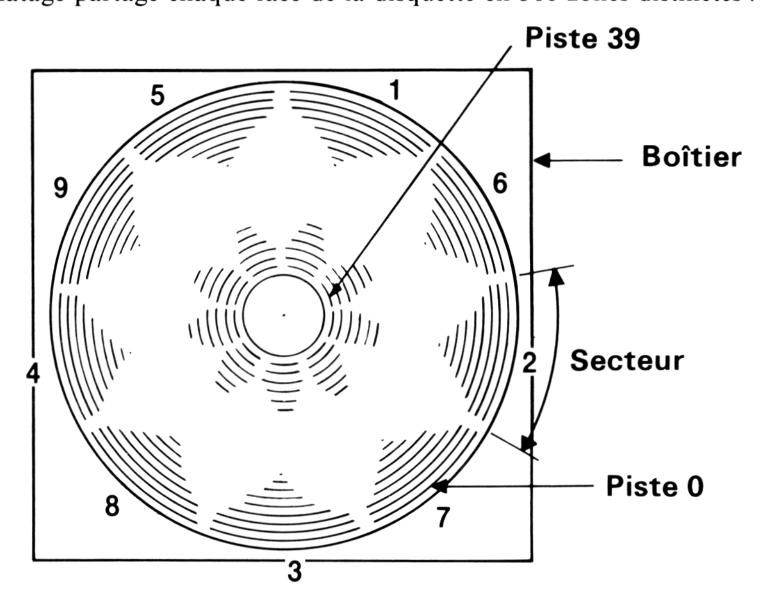
Vous ne pouvez pas utiliser une disquette vierge comme vous le feriez pour une cassette. La disquette doit tout d'abord être formatée.

Vous devez « nommer » vos fichiers disque correctement. Si les noms de fichiers sur cassette obéissent à des règles très floues, ceux sur disquette doivent au contraire se conformer strictement aux normes CP/M, détaillées plus loin dans ce chapitre.

Formatage des disquettes

Avant d'écrire des données sur une disquette vierge, il faut commencer par la formater. Le formatage peut être vu comme la construction d'une série d'étagères dotées de compartiments pour le stockage des informations - en d'autres mots, une structure capable de stocker et de restituer des données.

Le formatage partage chaque face de la disquette en 360 zones distinctes :



La disquette comporte 40 pistes, depuis la piste 0 (extérieure) jusqu'à la piste 39 (intérieure). Chaque piste se divise en 9 secteurs.

Chaque secteur, quelle que soit la piste, peut contenir 512 octets de données, soit 180 Koctets d'espace disponible sur chaque face de la disquette.

Premières étapes pour l'utilisation des disquettes système CP/M

Avant d'utiliser une disquette vierge pour l'écriture et la lecture de vos programmes, formatez-la à l'aide de la face 1 d'une des disquettes CP/M.

Allumez votre système et insérez la face 1 d'une des disquettes CP/M dans l'unité.

S'il s'agit d'un système à 2 unités, insérez la disquette CP/M dans l'unité principale (Unité A).

Puis tapez :

```
|cpm [RETURN]
```

(La barre verticale s'obtient avec les touches **[SHIFT]** et@).

Après quelques secondes, le message suivant apparaît sur l'écran :

```
CP/M Plus   Amstrad Consumer Electronics plc
```

Ce symbole indique que le système passe sous le contrôle de CP/M Plus.

Le symbole **A**> à l'écran est l'équivalent du **Ready** sous BASIC. Il indique que l'ordinateur est prêt à recevoir vos instructions.

Une fois sous CP/M, les instructions BASIC ne sont plus comprises par l'ordinateur.

Si vous tapez par exemple la commande BASIC :

```
cls [RETURN]
```

...l'ordinateur répond en répétant le mot incompris, suivi d'un point d'interrogation :

```
CLS?
```

...montrant ainsi qu'il ne comprend pas votre commande.

Abordons maintenant quelques commandes de CP/M. Tapez :

```
dir [RETURN]
```

Vous allez voir apparaître le catalogue des **COM**mandes et utilitaires du CP/M. L'un d'eux est **DISCKIT3**. Tapez :

```
disckit3
```

Au bout de quelques secondes, le message initial du DISC KIT s'affiche en haut de l'écran, suivi de :

```
One drive found
```

indiquant que vous exploitez l'utilitaire DISC KIT et que l'ordinateur a reconnu l'utilisation du système avec une seule unité de disquettes (celle qui est intégrée à l'ordinateur).

Si votre système comprend une unité supplémentaire, le message affiché sera :

```
Two drives found
```

Vous verrez alors au bas de l'écran :

Copy	7	
Format	4	
Verify	1	
Exit from program	Ø	

Il s'agit du menu principal du DISC KIT. Les numéros inscrits dans les cases correspondent aux touches de fonction situées à droite du clavier (**fØ**, **f1**, **f4** et **f7**). Faites votre sélection à l'aide de la touche voulue.

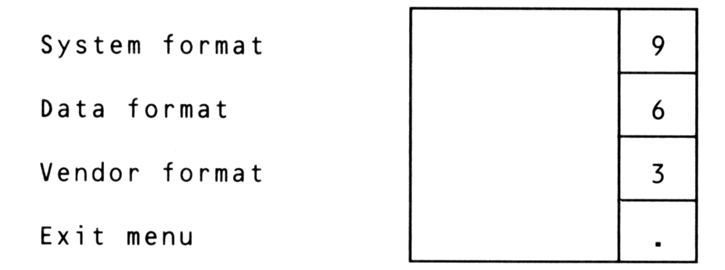
Appuyer sur la touche de fonction **fØ**, à ce stade des opérations, vous fait sortir du programme DISC KIT pour vous ramener sous le système CP/M (message **A** >).

Puisque nous voulons formater une disquette, appuyez sur **f4**.

ATTENTION

LE FORMATAGE D'UNE DISQUETTE DEJA ENREGISTREE ENTRAINERA L'EFFACEMENT DE SON CONTENU.

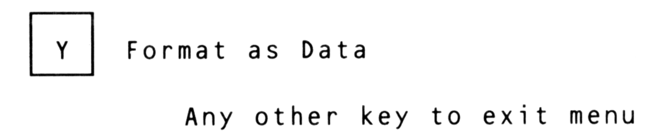
Un nouveau menu apparaît alors vous proposant plusieurs formats :



Comme précédemment, vous allez appuyer sur une des touches de fonctions (**f3**, **f6** ou **f9**) pour sélectionner le type de format souhaité. Nous reviendrons par la suite sur ces différents formats ; pour le moment, choisissons le format Data, soit la touche f6.

Appuyer sur la touche . , située sous **f3**, **f6** et **f9**, vous fait sortir du mode Format pour vous ramener au menu DISC KIT.

Après sélection de la touche de fonction 6 (en supposant que votre système ne possède qu'une seule unité de disquette), vous recevez à l'écran le message :



Vous devez alors retirer la disquette système CP/M et insérer la disquette concernée, la face à formater tournée vers le haut.

Tapez **Y** (pour « oui », à savoir, « poursuivre et formater la disquette »).

Le formatage de la disquette s'effectue, piste par piste (de 0 à 39), et le numéro de la piste en cours de formatage s'affiche dans l'angle supérieur gauche de l'écran.

Vous ne pouvez pas formater une disquette dont le trou de protection en écriture est ouvert. Si vous tentez de le faire, le message ci-dessous vous est adressé :

```
Disc write-protected
Insert disc to format
R-etry or C-ancel
```

... Tapez alors **C** pour annuler, retirez la disquette et insérez-en une autre formatable.

Veillez à ne jamais fermer le trou de protection en écriture d'une disquette dont vous souhaitez conserver le contenu. C'est NOTAMMENT le cas des disquettes système CP/M.

Une fois le formatage terminé, l'ordinateur vous demande de retirer la disquette et d'appuyer sur une touche quelconque du clavier pour poursuivre.

Après quoi vous pouvez insérer une autre disquette à formater et appuyer de nouveau sur **Y**. Vous avez la possibilité de répéter cette opération autant de fois que vous le désirez.

Après formatage de toutes vos disquettes, appuyez sur une touche quelconque (à l'exception de la touche **Y**) pour revenir au menu principal du DISC KIT.

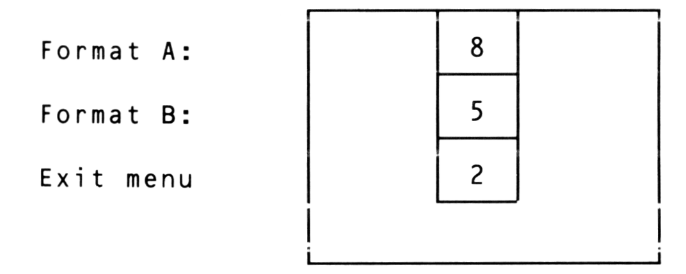
Nous aborderons les options **Copy** et **Verify** ultérieurement. Pour l'heure, maintenant que nous savons formater sous CP/M, nous allons réinitialiser l'ordinateur à l'aide des touches **[CONTROL]**, **[SHIFT]** et **[ESC]**.

Gardez toujours une copie de vos disquettes CP/M en lieu sûr car elles sont la clé de voûte de votre système. Nous vous montrerons par la suite comment faire une copie de travail de ces précieuses disquettes de façon à pouvoir les mettre de côté soigneusement.

Formatage des disquettes dans un système à deux unités

Suivez les instructions ci-dessus, sélectionnez l'option **Format** du menu principal du DISC KIT en appuyant sur la touche de fonction **f4**, puis sélectionnez **Data format** au moyen de la touche **f6**.

Vous verrez alors s'afficher un troisième menu d'options, portant sur le choix de l'unité à partir de laquelle vous procéderez au formatage :



La sélection de l'option **Format B** (touche **f5**), vous permettra de laisser votre disquette système dans l'unité A (face 1 vers le haut) ; vous placerez donc la disquette à formater dans l'unité B.

Tapez alors **Y** pour procéder au formatage, ou sur une autre touche pour revenir au menu principal du DISC KIT.

Si vous avez sélectionné l'option **Format A** (touche **f8**) vous DEVEZ retirer la disquette système de l'unité A et la remplacer par la disquette à formater.

N'OUBLIEZ JAMAIS DE PROTEGER LES DISQUETTES SYSTEME CONTRE UN ECRASEMENT ACCIDENTEL DES DONNEES QU'ELLES CONTIENNENT.

Maintenant que nous avons formaté une disquette vierge (ou deux), nous pouvons commencer à expérimenter les commandes de gestion des disquettes propres au BASIC.

Sauvegarde d'un programme sur disquette

Vous avez tapé un programme dans la mémoire de l'ordinateur. Vous voulez le sauvegarder sur disquette. Tapez :

```
save "nomfich" [RETURN]
```

Remarquez que vous devez donner un nom à votre programme.

Un nom de fichier sur disquette se compose de deux parties ou zones. La première, obligatoire, peut contenir jusqu'à 8 caractères, lettres ou nombres (les espaces et les signes de ponctuation sont interdits). Cette première zone contient généralement le nom du programme.

La deuxième zone, facultative, peut contenir jusqu'à trois caractères, mais cette fois encore ni espace ni signe de ponctuation. Ces deux parties sont séparées par un point.

En l'absence d'une deuxième zone spécifiée, le système en choisit une par défaut, **.BAS** pour les fichiers BASIC ou **.BIN** pour les fichiers binaires (en code machine).

Comme exemple de sauvegarde sur disquette, écrivez un petit programme dans la mémoire de l'ordinateur, insérez une disquette formatée dans l'unité et tapez :

```
save "exemple" [RETURN]
```

Après quelques secondes, le message **Ready** apparaît sur l'écran, le programme est maintenant sauvegardé sur la disquette. (Dans le cas contraire, vérifiez ce que dit le message d'erreur sur l'écran. La disquette peut ne pas avoir été insérée dans la bonne unité ou le trou de protection en écriture peut être ouvert, ou encore la commande avoir été mal écrite).

Catalogue

Après la sauvegarde du programme précédent, tapez :

```
cat [RETURN]
```

Vous verrez apparaıtre :

```
Drive A: user 0

EXEMPLE.BAS  1K

177K free
```

Le nom du programme sera affiché avec ses deux zones, la deuxième ayant été spécifiée ou non, ainsi que la longueur du programme (arrondie au Ko supérieur). L'espace libre total du disque sera également affiché.

Chargement d'un programme sur disquette

Les programmes sur disquette sont chargés dans la mémoire de l'ordinateur à l'aide de la commande :

```
load "nomfich" [RETURN]
run [RETURN]
```

...ou pour un démarrage immédiat :

```
run "nomfich" [RETURN]
```

Notez au passage que run est la seule commande permettant d'exécuter des programmes protégés.

| A et | B

Avec un système à deux unités de disquette, vous devez spécifier l'unité (A ou B) sur laquelle vous désirez exécuter une fonction. Tapez pour cela :

```
|a [RETURN]
```

...ou :

```
|b [RETURN]
```

...avant d'entrer les commandes **SAVE**, **CAT** ou **LOAD**.

Copie de programmes de disquette à disquette

En utilisant les commandes déjà abordées dans ce chapitre, vous constaterez que l'on peut copier un programme d'une disquette (source) sur une autre, en le chargeant d'abord dans la mémoire de l'ordinateur puis en le sauvegardant sur l'autre disquette (disquette destination).

Si vous utilisez un système à deux unités, vous pouvez insérer la disquette avec le programme à charger dans une unité, B par exemple, et la disquette qui reçoit le programme dans l'autre unité, A par conséquent ; ce qui donne :

```
|b [RETURN]
load "nomfich" [RETURN]
|a [RETURN]
save "nomfich" [RETURN]
```

Outre la méthode habituelle de sauvegarde de fichiers BASIC, vous disposez de quatre autres méthodes de sauvegarde de fichiers :

```
run "nomfich" [RETURN]
```

...il en existe trois autres destinées à des applications plus spécialisées.

Fichiers ASCII

```
save "nomfich",a [RETURN]
```

Le suffixe **,a** demande à l'ordinateur de sauvegarder le programme ou les données sous la forme d'un fichier de texte ASCII. Cette méthode de sauvegarde s'applique aux fichiers de traitement de texte et à d'autres programmes d'application. Son utilisation sera abordée ultérieurement avec les applications concernées.

Fichiers protégés

```
save "nomfich",p [RETURN]
```

Le suffixe **,p** demande à l'ordinateur de protéger le programme, c'est-à-dire d'interdire son listage après chargement ou son interruption, par la touche **[ESC]**, après démarrage.

Les programmes sauvegardés de cette façon ne peuvent être lancés que directement, à l'aide des commandes :

```
save "nomfich" [RETURN]
```

...ou :

```
chain "nomfich" [RETURN]
```

Si vous prévoyez de modifier un programme, il est indispensable d'en garder une copie sous forme non protégée, donc sans le suffixe « **,p** ».

Fichiers binaires

save "nomfich",b, <adresse de départ> , <nombre d'octets> [, < point d'entrée éventuel>] **[RETURN]**

Cette option permet d'effectuer une sauvegarde binaire qui provoque le stockage de blocs entiers de données de la RAM sur la disquette, tels qu'ils apparaissent dans la mémoire. Il est nécessaire dans ce cas d'indiquer à l'ordinateur le début de la zone mémoire à sauvegarder, sa longueur et, s'il s'agit d'un programme, l'adresse mémoire à partir de laquelle le fichier doit s'exécuter.

Vidage d'écran

Cette sauvegarde binaire permet de stocker directement sur disquette les données de la mémoire écran sous forme d'un vidage d'écran. Le contenu de l'écran est alors sauvegardé tel qu'il apparaît à l'affichage, à l'aide de la commande :

```
save "pagecran",b,49152,16384 [RETURN]
```

49152 est l'adresse de départ de la mémoire écran et **16384** indique la longueur de mémoire écran que vous désirez sauvegarder.

Pour rappeler ces données à l'écran, tapez :

```
load "pagecran" [RETURN]
```

Vous trouverez plus loin dans le manuel des informations complémentaires sur la manipulation des fichiers programme entre disquettes/cassettes.

Enfin, assurez-vous que vous avez bien respecté les avertissements indiqués dans la rubrique « IMPORTANT » :

REMARQUES RELATIVES A L'INSTALLATION : 5, 6 et 7.
REMARQUES RELATIVES AU FONCTIONNEMENT : 1, 2, 3, 4, 5, 6, 7 et 9.

Partie 8 : Modes couleurs et graphiques

L'ordinateur personnel Schneider 6128 dispose de trois modes d'affichage à l'écran : Mode 0, Mode 1 et Mode 2.

A la mise sous tension, l'ordinateur est automatiquement en mode 1.

Pour comprendre le système des modes, appuyez, après la mise en route, sur la touche numérique 1. Maintenez-la enfoncée jusqu'à obtenir deux lignes pleines de 1. Si vous comptez le nombre de 1 d'une ligne, vous verrez qu'il y en a 40 : en Mode 1, il y a 40 colonnes. Appuyez sur **[RETURN]**, vous aurez le message « **Syntax Error** ». Ne vous inquiétez pas, nous avons simplement pris le plus court chemin pour revenir au message **Ready**.

Tapez alors :

```
mode 0 [RETURN]
```

Vous voyez que les caractères sur l'écran sont plus larges. Appuyez sur le chiffre 1 pour avoir deux lignes pleines de 1. Vous en décomptez 20 par ligne. Il y a donc 20 colonnes en Mode 0. Appuyez sur **[RETURN]** de nouveau.

Et tapez :

```
mode 2 [RETURN]
```

L'écriture est plus petite dans ce mode ; une ligne de 1 en comprendra 80. Il y a donc 80 colonnes en Mode 2.

En résumé :

Mode 0 = 20 colonnes
Mode 1 = 40 colonnes
Mode 2 = 80 colonnes

Tapez **[RETURN]** une fois de plus.

Couleurs

Vous avez le choix entre 27 couleurs. Sur un moniteur vert, elles apparaissent en camaîeux de vert. Si vous disposez du moniteur vert GT65, vous pouvez acheter l'adaptateur Peritel pour vous servir de votre téléviseur couleur.

En Mode 0, on peut avoir 16 couleurs en même temps à l'écran.
En Mode 1, on peut en avoir 4.
En Mode 2, on peut en avoir 2.

Il est possible de changer la couleur de la bordure du cadre extérieur, appelée **BORDER**, la couleur de l'espace d'écriture appelé **PAPER** et la couleur du stylo, appelé **PEN** (c'est-à-dire la couleur des caractères), indépendamment les unes des autres.

Les 27 couleurs disponibles sont indiquées dans le Tableau 1, chacune avec le numéro de référence de la couleur (**INK** = encre).

Pour plus de commodité, ce tableau (version anglaise) figure en haut à droite de l'ordinateur.

Tableau des couleurs

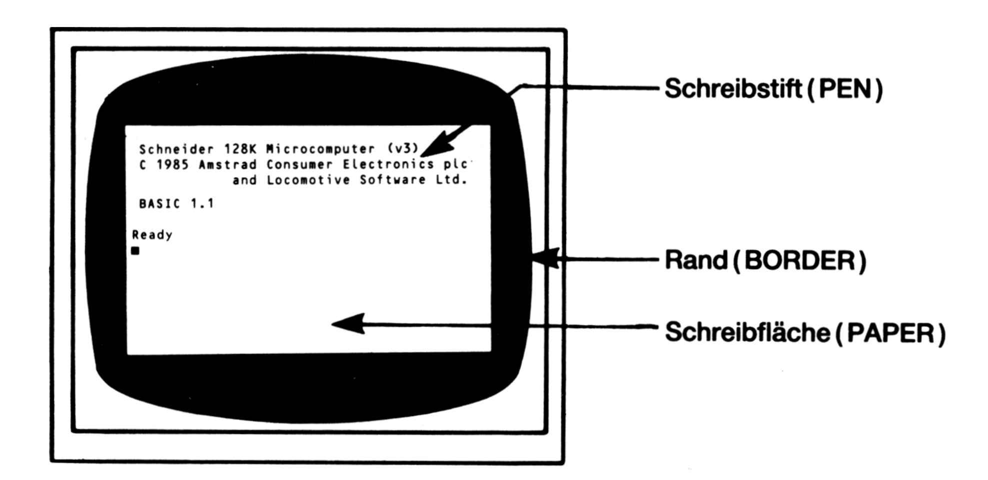
INK No	Couleur de l'encre	INK No	Couleur de l'encre
0	Noir	14	Bleu Pastel
1	Bleu	15	Orange
2	Bleu Vif	16	Rose
3	Rouge	17	Magenta Pastel
4	Magenta	18	Vert Vif
5	Mauve	19	Vert Marin
6	Rouge Vif	20	Turquoise Vif
7	Violet	21	Vert Citron
8	Magenta Vif	22	Vert Pastel
9	Vert	23	Turquoise Pastel
10	Turquoise	24	Jaune Vif
11	Bleu Ciel	25	Jaune Pastel
12	Jaune	26	Blanc Brillant
13	Blanc		

Tableau 1 : Les numéros d'encre (INK) et les couleurs

Comme nous le disions auparavant, l'ordinateur est en mode 1 à l'origine. Pour y revenir à partir d'un autre mode, tapez :

```
mode 1 [RETURN]
```

Configuration de l'écran



BORDER est le cadre entourant la zone d'écriture ou **PAPER**. (Au départ, **BORDER** et **PAPER** sont bleus tous les deux). Les caractères affichés apparaissent toujours à l'intérieur du cadre. **PAPER** désigne le fond sur lequel ils s'affichent. Le stylo (**PEN**) colore les caractères.

Nous allons maintenant aborder la sélection des couleurs affichées à l'écran et la façon de les modifier.

A la mise sous tension ou lors d'une réinitialisation, le cadre est toujours de la couleur numéro 1. En consultant le tableau 1, vous constaterez que c'est la couleur bleue. Pour modifier la couleur du cadre, utilisez la commande : **BORDER** suivie du numéro de la couleur sélectionnée. Pour un cadre blanc, tapez :

```
border 13 [RETURN]
```

Pour l'instant, pas de problème. Compliquons un peu l'exercice...

A la mise sous tension ou à la réinitialisation, les numéros **PAPER** et **PEN** sont automatiquement sélectionnés : **PAPER = Ø**, **PEN = 1**. Mais ceci ne veut PAS dire pour autant que les couleurs du papier et du stylo sont celles du tableau 1...

En effet, **PAPER Ø** et **PEN 1** ne sont pas des paramètres de couleur. Imaginez que vous disposiez de 4 stylos numérotés de 0 à 3 et que, pour chacun d'entre eux, vous ayez le choix entre 27 couleurs d'encre, numérotées de 0 à 26. Vous voyez donc que le stylo numéro 1 peut prendre plusieurs couleurs. Libre à vous, bien entendu, de choisir la même couleur d'encre pour les quatre stylos.

C'est ce qui se passe avec l'ordinateur. A l'aide des commandes **PEN** et **INK**, vous pouvez sélectionner le stylo qui vous intéresse et sa couleur d'encre.

En vous rappelant que vous opérez en Mode 1 (40 colonnes), consultez le tableau 2 ci-dessous et vous verrez dans la première et la troisième colonne que le stylo numéro 1 correspond à la couleur numéro 24. Si vous vous reportez au tableau 1, vous constaterez que la couleur numéro 24 est jaune vif ; c'est bien la couleur des caractères affichés à la mise sous tension.

Couleurs par défaut

N° de Papier/stylo	Mode 0 Couleur d'encre	Mode 1 Couleur d'encre	Mode 2 Couleur d'encre
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	clignotement1,24	20	1
15	clignotement16,11	6	24

Tableau 2. Référence PAPER/PEN/MODE/INK

Attention ! Les correspondances **PAPER/PEN/INK** indiquées au tableau 2 ne sont pas fixes. En effet, il ne s'agit là que des couleurs par défaut, sélectionnées à la mise sous tension et à la réinitialisation de l'ordinateur. Vous pouvez les modifier à l'aide de la commande **INK**, constituée de deux paramètres, le premier pour le numéro du papier ou du stylo et le second pour le numéro de la couleur d'encre. Une virgule sépare ces deux paramètres.

Sachant que le stylo numéro 1 est sélectionné, nous allons changer sa couleur d'encre pour obtenir des caractères oranges.

Tapez :

```
ink 1,15 [RETURN]
```

...les caractères affichés ont effectivement changé de couleur.

La commande **INK** sert aussi à modifier la couleur du fond. Le premier numéro **Ø** est sélectionné à la mise sous tension ; nous allons modifier sa couleur pour obtenir un papier numéro **Ø** de couleur verte (numéro **9**) :

```
ink 0,9 [RETURN]
```

Changeons maintenant de stylo :

```
pen 3 [RETURN]
```

Vous remarquerez alors que seuls les nouveaux caractères (tapés après la commande) changent de couleur. Avec le stylo numéro **3**, la couleur d'encre initiale est rouge vif (numéro **6**, tableau 2). Passons à la couleur rose en tapant :

```
ink 3,16 [RETURN]
```

3 est le paramètre du stylo sélectionné précédemment par la commande : **pen 3**, et **16** correspond à la couleur d'encre rose.

Changeons maintenant le papier (fond sur lequel s'afficheront les caractères). Lorsque vous sélectionnez un nouveau papier, la couleur du fond ne change que pour les caractères affichés après l'exécution de cette commande. Tapez :

```
paper 2 [RETURN]
```

Cette fois encore, vérifiez sur les tableaux 1 et 2 que le numéro **2** de **PAPER** correspond à un fond turquoise vif. Colorez-le en noir en tapant :

```
ink 2,0 [RETURN]
```

L'écran affiche maintenant des caractères écrits avec les stylos 1 et 3 sur des fonds **0** et **2**. Vous pouvez sélectionner des couleurs **INK** de stylo et papier, que vous n'êtes pas en train d'utiliser. Tapez par exemple :

```
ink 1,2 [RETURN]
```

Les caractères déjà tapés avec le stylo **1** changent de couleur.

Tapez :

```
cls [RETURN]
```

...pour effacer l'écran.

Vous devez être à même désormais de revenir aux couleurs d'origine (cadre et fond bleus, caractères jaunes vif) à l'aide des commandes **BORDER**, **PAPER**, **PEN** et **INK**. Essayez. Si vous n'y parvenez pas, réinitialisez l'ordinateur à l'aide des touches **[CONTROL]**, **[SHIFT]** et **[ESC]**.

Couleurs clignotantes (Flash)

On peut faire alterner les caractères d'une couleur à l'autre en utilisant un autre numéro de couleur pour la définition **INK** de la commande **PEN**.

La couleur des caractères va changer rapidement entre le blanc brillant et le rouge vif. Réinitialisez à l'aide des touches **[CONTROL]**, **[SHIFT]** **[ESC]** et tapez :

```
ink 1,26,6 [RETURN]
```

1 correspond au numéro du stylo (**PEN**), **26** au blanc brillant et **6** à la couleur qu'il va prendre, rouge vif.

Il est également possible de faire clignoter le fond **PAPER** d'une couleur à l'autre. Vous devez alors ajouter à la commande **INK** un numéro de couleur supplémentaire.

Pour faire clignoter **PAPER** entre vert et jaune vif, tapez :

```
ink 0,9,24 [RETURN]
```

Ø correspond au numéro du papier **PAPER**, **9** au vert et **24** au jaune vif.

Réinitialisez à l'aide des touches **[CONTROL] [SHIFT] [ESC]**.

Le tableau 2 vous montre qu'en mode **Ø**, deux des numéros de **PEN** (numéros **14** et **15**) ou de **PAPER** (numéros **14** et **15**) représentent les couleurs clignotantes d'origine. Autrement dit, leurs commandes **INK** ont été préprogrammées avec un paramètre de couleur supplémentaire.

On le constate en faisant :

```
mode 0 [RETURN]
pen 15 [RETURN]
```

...afin que le mot **Ready** alterne entre bleu ciel et rose ! Si vous faites ensuite :

```
paper 14 [RETURN]
cls [RETURN]
```

...outre **Ready** clignotant entre bleu ciel et rose, le fond de **PAPER** alterne maintenant entre jaune et bleu.

Les numéros de stylo et papier **14** et **15** peuvent être reprogrammés par la commande **INK**, pour obtenir d'autres couleurs intermittentes ou une couleur fixe.

Enfin, on peut faire clignoter la couleur du cadre en ajoutant un autre numéro de couleur à la commande **BORDER**.

Par exemple :

```
border 6,9 [RETURN]
```

La couleur du cadre alterne du rouge vif au vert. Pour le cadre, vous avez le choix entre une ou deux couleurs parmi les 27 disponibles, quel que soit le mode de fonctionnement 0, 1 ou 2.

Réinitialisez l'ordinateur à l'aide des touches **[CONTROL] [SHIFT] [ESC]**.

Pour voir ce qu'il est possible de faire avec les couleurs, tapez ce programme et lancez-le (**run**).

```
10 MODE 0 [RETURN]
20 vitesse=600:REM Determine la vitesse du programme [RETURN]
30 FOR b=0 TO 26 [RETURN]
40 LOCATE 1,12 [RETURN]
50 BORDER b [RETURN]
60 PRINT"COULEUR DU CADRE";b [RETURN]
70 FOR t=1 TO vitesse [RETURN]
80 NEXT t,b [RETURN]
90 CLG [RETURN]
100 FOR p=0 TO 15 [RETURN]
110 PAPER p [RETURN]
120 PRINT"PAPIER";p [RETURN]
130 FOR n=0 TO 15 [RETURN]
140 PEN n [RETURN]
150 PRINT"STYLO";n [RETURN]
160 NEXT n [RETURN]
170 FOR t=1 TO vitesse*2 [RETURN]
180 NEXT t,p [RETURN]
190 MODE 1 [RETURN]
200 BORDER 1 [RETURN]
210 PAPER 0 [RETURN]
220 PEN 1 [RETURN]
230 INK 0,1 [RETURN]
240 INK 1,24 [RETURN]
run [RETURN]
```

IMPORTANT

Dans ce programme, comme dans les chapitres suivants et les listages du manuel, les mots clés du BASIC apparaissent en majuscules. Il s'affichent ainsi sur la demande d'une instruction **LIST**. Il est généralement préférable de taper les instructions et les programmes en minuscules pour mieux détecter les erreurs de frappe lors des listages des programmes (les mots clés mal orthographiés ne sont alors pas convertis en majuscules).

Jusqu'à la fin de ce cours élémentaire, les programmes sont listés en majuscules et en minuscules afin de vous familiariser avec ce procédé.

Un nom de variable, **x** ou **$** par exemple, ne sera pas converti en majuscule lors du **LIST**age du programme, bien qu'il soit reconnu par le programme de toute façon.

Attention

A partir de maintenant, nous ne préciserons plus de taper **[RETURN]** à la fin de chaque ligne. Nous supposons que vous en avez pris le réflexe.

Graphiques

Il y a un certain nombre de caractères dans la mémoire de l'ordinateur. Pour afficher l'un de ces caractères, on utilise le mot réservé :

```
chr$( )
```

Entre les parenthèses, on entre le numéro du symbole (entre 32 et 255).

Faites **[CONTROL]**, **[SHIFT]** et **[ESC]** pour la remise à zéro, puis tapez :

```
print chr$(250)
```

N'oubliez pas d'appuyer sur **[RETURN]**. Sur l'écran apparaît le caractère N° **250**, qui représente un homme marchant vers la droite.

Pour afficher tous les caractères et symboles avec leurs numéros, tapez le programme suivant, en n'oubliant pas d'appuyer sur **[RETURN]** après chaque ligne.

```
10 FOR n=32 TO 255
20 PRINT n;CHR$(n);
30 NEXT n
run
```

Vous trouverez au chapitre « Pour information... » tous les caractères accompagnés de leurs numéros.

LOCATE (place le curseur)

On se sert de cette commande pour mettre le curseur à un certain endroit de l'écran. Sans la commande LOCATE, le curseur se trouve en haut et à gauche de l'écran, ce qui correspond en coordonnées x,y à 1,1 (x étant la position horizontale, y la position verticale). En Mode 1, il y a 40 colonnes et 25 lignes. Pour placer un caractère en haut au milieu de l'écran, il faut indiquer x = 20 et y = 1. Essayons :

mode 1 ...l'écran s'efface ; le curseur se trouve en haut, à gauche.

```
10 LOCATE 20,1
20 PRINT CHR$(250)
run
```

Pour vérifier qu'il s'agit bien du haut de l'écran, tapez :

```
border 0
```

Le cadre sera noir et vous verrez le petit bonhomme au milieu de la ligne supérieure.

En Mode 0, il n'y a que 20 colonnes, mais toujours 25 lignes. Si vous tapez :

```
mode 0
run
```

...le bonhomme apparaît en haut à droite de l'écran, car la 20ème colonne est la dernière du mode 0.

En mode 2, il y a 80 colonnes et 25 lignes. A l'aide du même programme, vous devinez certainement où va apparaître le petit bonhomme. Tapez :

```
mode 2
run
```

Revenez au Mode 1, en faisant :

```
mode 1
```

A vous maintenant de faire varier les nombres après **locate** et dans **CHR$** ( ). Pour l'exemple, essayez :

```
locate 20,12:print chr$(240)
```

Vous voyez une flèche au centre de l'écran. Dans cette instruction :

20 est la coordonnée horizontale (comprise entre 1 et 40)
12 est la coordonnée verticale (comprise entre 1 et 25)
240 est le numéro du caractère (compris entre 32 et 255)

Pour voir le caractère 250 répété sur l'écran, il suffit de taper :

```
10 CLS
20 FOR X=1 TO 39
30 LOCATE X,20
50 PRINT CHR$(250)
60 NEXT X
70 GOTO 10
run
```

Faites **[ESC]** deux fois pour interrompre le déroulement.

Pour effacer le caractère précédent avant d'afficher le suivant, tapez :

```
50 PRINT " ";CHR$(250)
```

(La nouvelle ligne **50** remplace automatiquement la précédente).

Maintenant, tapez :

```
run
```

FRAME

Pour que le mouvement du caractère soit plus joli, ajoutez la ligne suivante :

```
40 FRAME
```

La commande **FRAME** synchronise le mouvement de l'objet affiché sur la fréquence de balayage de la trame d'affichage. Si cette notion vous paraît trop technique, rappelez-vous simplement que cette commande sert à déplacer harmonieusement sur l'écran des caractères ou des graphiques.

On peut encore améliorer ce programme en ajoutant des petites pauses et en utilisant d'autres caractères.

Pour ce faire, tapez :

```
list
```

Puis ajouter les lignes suivantes au programme :

```
70 FOR n=1 TO 300:NEXT n
80 FOR x=39 TO 1 STEP -1
90 LOCATE x,20
100 FRAME
110 PRINT CHR$(251);" "
120 NEXT x
130 FOR n=1 TO 300:NEXT n
140 GOTO 20
run
```

PLOT (détermine un point sur l'écran)

Contrairement à la commande **LOCATE**, **PLOT** sert à fixer la position du curseur graphique, utilisant les coordonnées des pixels (un pixel est la plus petite unité de l'écran).

Remarque : le curseur graphique, différent du curseur des caractères, n'est pas visible.

Il y a 640 pixels horizontaux sur 400 verticaux. Les coordonnées x et y sont définies par rapport au coin inférieur gauche de l'écran, qui a pour coordonnées 0,0. Contrairement à la commande **LOCATE**, appliquée aux caractères, ces coordonnées sont les mêmes en Mode 0, 1 ou 2.

L'ordinateur ayant été remis à zéro à l'aide de **[CONTROL]** **[SHIFT]** et **[ESC]**, essayez :

```
plot 320,200
```

Un petit point apparaît au centre de l'écran.

Changeons de Mode en tapant :

```
mode 0
plot 320,200
```

Le point se trouve toujours au milieu de l'écran, mais agrandi. Changez à nouveau de mode et tapez la même commande pour voir le résultat en mode 2 :

```
mode 2
plot 320,200
```

Le point est toujours au centre, mais beaucoup plus petit.

Essayez de dessiner des points sur l'écran avec les différents modes pour vous habituer à la commande **PLOT**. Revenez ensuite en Mode 1 et videz l'écran en tapant :

```
mode 1
```

DRAW (Dessine une ligne)

Remise à zéro : **[CONTROL]** **[SHIFT]** et **[ESC]**. La commande **DRAW** dessine une ligne à partir de la position du curseur graphique. Pour mieux comprendre, dessinons un rectangle sur l'écran avec le programme suivant.

On commence par placer le curseur graphique avec la commande **PLOT**. Puis on dessine une ligne à partir de ce point, vers le coin supérieur gauche, puis vers la droite, etc... Tapez :

```
5 CLS
10 PLOT 10,10
20 DRAW 10,390
30 DRAW 630,390
40 DRAW 630,10
50 DRAW 10,10
60 GOTO 60
run
```

Actionnez **[ESC]** deux fois pour sortir du programme.

(Dans ce programme, l'ordinateur boucle indéfiniment à la ligne **60** jusqu'à ce que vous l'interrompiez en appuyant deux fois sur la touche **[ESC]**. Ce type d'instruction évite l'interruption automatique du programme après la dernière ligne et l'affichage du message **Ready**).

Ajoutez maintenant les lignes suivantes pour dessiner un deuxième rectangle à l'intérieur du premier :

```
60 PLOT 20,20
70 DRAW 20,380
80 DRAW 620,380
90 DRAW 620,20
100 DRAW 20,20
110 GOTO 110
run
```

Appuyez sur deux fois **[ESC]** pour sortir du programme.

MOVE

La commande **MOVE** fonctionne comme **PLOT**, c'est-à-dire que le curseur graphique se place sur le point des coordonnées x,y sans tracer le pixel au nouvel emplacement du curseur.

Tapez :

```
cls
move 639,399
```

Bien que nous ne le voyons pas à l'écran, le curseur graphique se trouve maintenant dans l'angle supérieur droit.

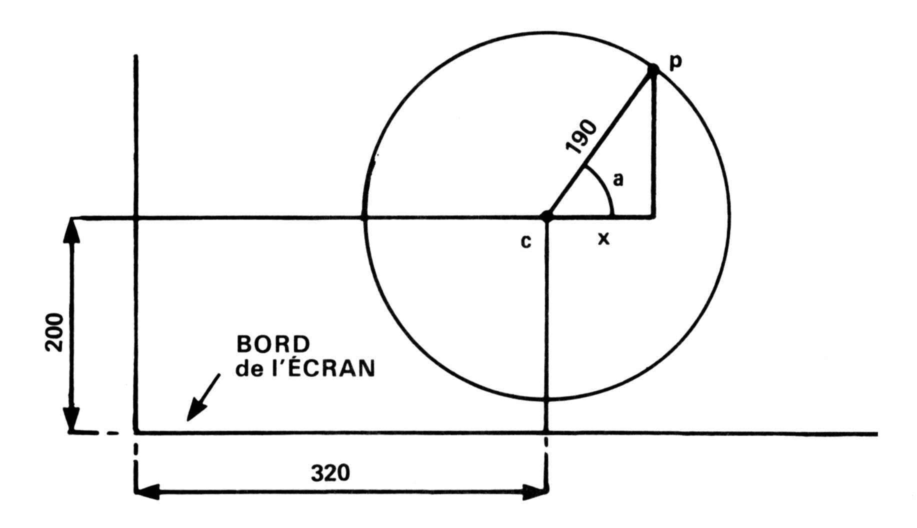
Mettons cette position en évidence en traçant une ligne à partir de ce point vers le centre de l'écran. Tapez :

```
draw 320,200
```

Cercles

On peut dessiner des cercles à l'aide des commandes PLOT ou DRAW. Pour dessiner un cercle, vous pouvez placer les coordonnées x et y de chaque point de sa circonférence. Sur la figure suivante, un point « p » peut être placé avec la commande PLOT, en utilisant les coordonnées x et y.

```
x=190*cos(a)
y=190*sin(a)
```



Positionnement du centre d'un cercle

Dans les programmes précédents, on avait fixé le centre par rapport au coin inférieur gauche de l'écran. Pour tracer un cercle au milieu de l'écran, il faut placer son centre aux coordonnées 320 et 200, puis dessiner tous les points du cercle par rapport à ce centre en ajoutant les coordonnées de ce dernier. Voici un programme pour obtenir un cercle :

```
new
10 CLS
20 DEG
30 FOR a=1 TO 360
40 MOVE 320,200
50 DRAW 320+190*COS(a),200+190*SIN(a)
60 NEXT
run
```

Le mot clé **NEW** précédant l'entrée du programme remet la mémoire à zéro (comme une réinitialisation), sans toutefois effacer l'écran.

Le rayon du cercle peut être réduit en diminuant le nombre 190 (nombre de pixels).

Pour obtenir un cercle déterminé d'une manière différente (en radians), enlevez la ligne **20** en tapant :

```
20
```

Pour dessiner un cercle plein (un disque) avec des lignes tracées à partir du centre, modifiez (**EDIT**) la ligne **50** en remplaçant le mot **PLOT** par le mot **DRAW**. Elle devient :

```
50 DRAW 320+190*COS(a),200+190*SIN(a)
```

Essayez maintenant avec et sans la ligne **20**.

Notez que dans la ligne **60** de ce programme, **NEXT** n'est pas suivi de **a**. On peut mettre seulement **NEXT** ; l'ordinateur détermine alors à quel **FOR** il se rapporte. Dans les programmes contenant de nombreux **FOR** et **NEXT**, il est toutefois préférable d'ajouter la variable après **NEXT** afin de mieux s'y retrouver.

ORIGIN

Dans le programme précédent, on s'est servi de la commande **MOVE** pour fixer le centre du cercle, puis on a ajouté les coordonnées x,y à celles du centre. Au lieu de cela, nous pouvons utiliser la commande **ORIGIN** (attention, pas de **E** à **ORIGIN**, c'est un mot réservé). Elle place les coordonnées x et y de chaque point en fonction d'**ORIGIN**. Vérifions ceci avec le programme :

```
new
10 CLS
20 FOR a=1 TO 360
30 ORIGIN 320,200
40 PLOT 190*COS(a),190*SIN(a)
50 NEXT
run
```

Pour obtenir quatre petits cercles sur l'écran, entrons le programme suivant :

```
new
10 CLS
20 FOR a=1 TO 360
30 ORIGIN 196,282
40 PLOT 50*COS(a),50*SIN(a)
50 ORIGIN 442,282
60 PLOT 50*COS(a),50*SIN(a)
70 ORIGIN 196,116
80 PLOT 50*COS(a),50*SIN(a)
90 ORIGIN 442,116
100 PLOT 50*COS(a),50*SIN(a)
110 NEXT
run
```

Pour expérimenter une autre méthode de création d'un cercle, tapez le programme :

```
new
10 MODE 1
20 ORIGIN 320,200
30 DEG
40 MOVE 0,190
50 FOR a=0 TO 360 STEP 10
60 DRAW 190*SIN(a),190*COS(a)
70 NEXT
run
```

Cette fois, une ligne est tracée (**DRAW**) de coordonnée en coordonnée sur l'ensemble de la circonférence. Le traçage est nettement plus rapide que le positionnement des points.

Une fois encore, observez la conséquence du retrait de la commande **DEG** par suppression de la ligne **30** et relancez le programme.

FILL

La commande **FILL** sert à remplir une zone d'écran délimitée par une courbe, le bord de l'écran, ou d'une fenêtre graphique.

Réinitialisez à l'aide des touches **[CONTROL]** **[SHIFT]** et **[ESC]**, puis tapez :

```
new
10 CLS
20 MOVE 20,20
30 DRAW 620,20
40 DRAW 310,380
50 DRAW 20,20
run
```

Un triangle apparaît à l'écran. Placez le curseur graphique au centre de l'écran en tapant :

```
move 320,200
```

A l'aide du mot clé **FILL** suivi d'un numéro de stylo (3 par exemple), remplissez l'écran à partir de la position du curseur graphique (centre de l'écran) jusqu'aux limites tracées. Tapez :

```
fill 3
```

Amenez maintenant le curseur graphique à l'extérieur du triangle :

```
move 0,0
```

Observez ce qui se passe si vous tapez :

```
fill 2
```

A l'aide du stylo numéro **2**, l'ordinateur a rempli la zone comprise entre les lignes tracées et les bords de l'écran.

Modifiez le programme et regardez ce qui se passe :

```
50 DRAW 50,50
60 MOVE 320,200
70 FILL 3
run
```

Toute interruption de lignes sur l'écran laisse « passer » la couleur du stylo.

Ce phénomène est illustré par le remplissage d'un cercle positionné à l'écran, puis d'un cercle tracé. Tapez :

```
new
10 CLS
20 FOR a=1 TO 360
30 ORIGIN 320,200
40 PLOT 190*COS(a),190*SIN(a)
50 NEXT
60 MOVE -188,0
70 FILL 3
run
```

Tapez maintenant :

```
new
10 MODE 1
20 ORIGIN 320,200
30 DEG
40 MOVE 0,190
50 FOR d=0 TO 360 STEP 10
60 DRAW 190*COS(d),190*SIN(d)
70 NEXT
80 MOVE -188,0
90 FILL 3
run
```

Vous pouvez rendre la circonférence du cercle invisible en utilisant la même couleur pour le stylo et le papier. Ajoutez cette ligne :

```
45 GRAPHICS PEN 2:INK 2,1
run
```

La commande **GRAPHICS PEN** sélectionne le stylo pour le traçage des graphiques. La commande **INK** spécifie la couleur d'encre de ce stylo qui, en l'occurrence, est la même que celle du papier (ici, numéro de couleur **1**).

Pour terminer, tapez ce programme de démonstration :

```
10 MODE 0:BORDER 13
20 MOVE 0,200:DRAW 640,200
30 FOR x=80 TO 560 STEP 80
40 MOVE x,0:DRAW x,400
50 NEXT:MOVE -40,300
60 FOR c=0 TO 7
70 MOVER 80,0:FILL c
80 MOVER 0,-200:FILL c+8
90 MOVER 0,200:NEXT
100 GOTO 100
run
```

Les couleurs des zones en plein peuvent être modifiées après leur remplissage. Tapez :

```
100 SPEED INK 30,30
110 BORDER RND*26
120 INK RND*15,RND*26,RND*26
130 FOR t=1 TO 500:NEXT:GOTO 110
run
```

Compléments d'information...

Consultez la partie « Graphiques » du chapitre « A vos heures de loisir... » pour un guide complet des possibilités graphiques du CPC664.

Pour terminer ce chapitre, voici quelques programmes de démonstration graphique intégrant des commandes et des méthodes désormais à votre portée. Chacun de ces programmes trace des motifs en continu.

```
new
10 BORDER 0:GRAPHICS PEN 1
20 m=CINT (RND*2):MODE m
30 i1=RND*26:i2=RND*26
40 IF ABS(i1-i2)<10 THEN 30
50 INK 0,i1:INK 1,i2
60 s=RND*5+3:ORIGIN 320,-100
70 FOR x=-1000 TO 0 STEP s
80 MOVE 0,0:DRAW x,300:DRAW 0,600
90 MOVE 0,0:DRAW -x,300:DRAW 0,600
100 NEXT:FOR t=1 TO 2000:NEXT :GOTO 20
run
```

```
10 MODE 1:BORDER 0:PAPER 0
20 GRAPHICS PEN 2:INK 0,0
30 EVERY 2200 GOSUB 150
40 FLAG=0:CLG
50 INK 2,14+RND*12
60 B%=RND*5+1
70 C%=RND*5+1
80 ORIGIN 320,200
90 FOR A=0 TO 1000 STEP PI/30
100 X%=100*COS(A)
110 MOVE X%,X%
120 DRAW 200*COS(A/B%),200*SIN(A/C%)
130 IF FLAG=1 THEN 40
140 NEXT
150 FLAG=1:RETURN
run
```

```
10 MODE 1:BORDER 0:DEG
20 PRINT"VEUILLEZ PATIENTER "
30 FOR N=1 TO 3
40 INK 0,0:INK 1,26:INK 2,6:INK 3,18
50 IF N=1 THEN SA=120
60 IF N=2 THEN SA=135
70 IF N=3 THEN SA=150
80 IF N=1 THEN ORIGIN 0,-50,0,640,0,400 ELSE ORIGIN 0,0,0,64
0,0,400
90 DIM CX(5),CY(5),R(5),LC(5)
100 DIM NP(5)
110 DIM PX%(5,81),PY%(5,81)
120 ST=1:CX(1)=320:CY(1)=200:R(1)=80
130 FOR ST=1 TO 4
140 R(ST+1)=R(ST)/2
150 NEXT ST
160 FOR ST=1 TO 5
170 LC(ST)=0:NP(ST)=0
180 NP(ST)=NP(ST)+1
190 PX%(ST,NP(ST))=R(ST)*SIN(LC(ST))
200 PY%(ST,NP(ST))=R(ST)*COS(LC(ST))
210 LC(ST)=LC(ST)+360/R(ST)
220 IF LC(ST)<360 THEN 180
230 PX%(ST,NP(ST)+1)=PX%(ST,1)
240 PY%(ST,NP(ST)+1)=PY%(ST,1)
250 NEXT ST
260 CLS:CJ=RESTE(1):CJ=RESTE(2)
270 CJ=RESTE(3):INK 1,2:ST=1
280 GOSUB 350
290 LOCATE 1,1
300 EVERY 25,1 GOSUB 510
310 EVERY 15,2 GOSUB 550
320 EVERY 5,3 GOSUB 590
330 ERASE CX,CY,R,LC,NP,PX%,PY%:NEXT
340 GOTO 340
350 CX%=CX(ST):CY%=CY(ST):LC(ST)=0
360 FOR X%=1 TO NP(ST)
370 MOVE CX%,CY%
380 DRAW CX%+PX%(ST,X%),CY%+PY%(ST,X%),1+(ST MOD 3)
390 DRAW CX%+PX%(ST,X%+1),CY%+PY%(ST,X%+1),1+(ST MOD 3)
400 NEXT X%
410 IF ST=5 THEN RETURN
420 LC(ST)=0
430 CX(ST+1)=CX(ST)+1.5*R(ST)*SIN(SA+LC(ST))
440 CY(ST+1)=CY(ST)+1.5*R(ST)*COS(SA+LC(ST))
450 ST=ST+1
460 GOSUB 350
470 ST=ST-1
480 LC(ST)=LC(ST)+2*SA
490 IF (LC(ST) MOD 360)<>0 THEN 430
500 RETURN
```

```
510 IK(1)=1+RND*25
520 IF IK(1)=IK(2) OR IK(1)=IK(3) THEN 510
530 INK 1,IK(1)
540 RETURN
550 IK(2)=1+RND*25
560 IF IK(2)=IK(1) OR IK(2)=IK(3) THEN 550
570 INK 2,IK(2)
580 RETURN
590 IK(3)=1+RND*25
600 IF IK(3)=IK(1) OR IK(3)=IK(2) THEN 590
610 INK 3,IK(3)
620 RETURN
```

Partie 9 : Utilisation des sons

Les effets sonores sont produits par un haut-parleur intégré à l'ordinateur. Si vous utilisez l'adaptateur Péritel et un poste TV, baissez le volume au minimum.

Le niveau sonore est réglé par le bouton VOLUME, placé sur le côté droit de l'ordinateur. Le son peut aussi passer de la prise STEREO de l'ordinateur à l'entrée auxiliaire de votre chaîne hi-fi. Vous pouvez ainsi écouter vos programmes sonores en stéréo sur vos haut-parleurs ou votre casque d'écoute. Consultez la partie 2 du Cours Elémentaire pour connecter le casque à la prise stéréo de l'ordinateur.

La commande SOUND

La commande du son, **SOUND**, comporte sept paramètres. Si les deux premiers doivent être précisés, les autres sont facultatifs. La commande est écrite de la manière suivante :

> **SOUND** <état de canal>, <periode sonore>, <durée>, <volume>, <enveloppe de volume>, <enveloppe de ton>, <bruitage>.

D'apparence complexe, ces paramètres sont en fait faciles à identifier. Nous allons les étudier un par un.

Etat de canal

Pour plus de simplicité, nous allons provisoirement l'assimiler au canal sonore lui-même. Il existe trois canaux sonores et nous nous occuperons pour l'instant de l'<état de canal> numéro 1.

Période sonore

La <période sonore> concerne la définition technique de la hauteur de note : do, ré, mi, fa, sol, la, si. Chaque note est associée à un nombre représentant sa <période sonore>. En vous reportant au chapitre « Pour information... », vous verrez que la note do correspond à une période de **239**.

Réinitialisez l'ordinateur **[CONTROL]** **[SHIFT]** et **[ESC]** puis tapez :

```
10 SOUND 1,239
run
```

Vous entendrez un do pendant 0,2 seconde.

Si vous ne percevez aucun son, assurez-vous que le volume de votre ordinateur n'est pas sur zéro. Tapez **RUN** une nouvelle fois.

Durée

Ce paramètre détermine la durée du son et s'exprime en centièmes de seconde. En l'absence de spécification, l'ordinateur fixe un chiffre de **2Ø**, soit 0,01 x 20, ce qui donne bien une durée de 0,2 seconde.

Pour faire durer la note une seconde, il faudra donc lui attribuer un paramètre de **1ØØ** ; pour 2 secondes, **2ØØ**. Tapez :

```
10 SOUND 1,239,200
run
```

La même note do dure 2 secondes.

Volume

Ce paramètre détermine le volume de départ d'une note. Ce nombre varie de **Ø** à **15**. **Ø** correspond au volume minimum et **15** au volume maximum. En l'absence de spécification, l'ordinateur fixe un chiffre de **12**. Tapez :

```
10 SOUND 1,239,200,5
run
```

Remarquez le volume du son, puis tapez un numéro de volume supérieur :

```
10 SOUND 1,239,200,15
run
```

Vous devez obtenir un son plus fort.

Enveloppe de volume

Pour faire varier le volume d'un son pendant son émission, vous pouvez spécifier une enveloppe de volume à l'aide de la commande **ENV**. Vous avez la possibilité de fixer plusieurs <enveloppes de volume> différentes, chacune ayant son propre numéro de référence, tout comme la commande **SOUND**.

Si vous avez créé une enveloppe de référence N° **1** et que vous désiriez l'utiliser dans une commande **SOUND**, tapez **1** à l'endroit requis pour le paramètre de l'enveloppe. Nous expliquerons bientôt la création d'une enveloppe de volume.

Enveloppe de tonalité

Vous pouvez utiliser la commande **ENT** pour faire varier le ton d'une note pendant son émission. Vous avez la possibilité de fixer plusieurs enveloppes de tonalité qui, comme la commande **SOUND**, ont chacune un numéro de référence. Si vous créez une enveloppe de tonalité de **1** et que vous faites appel à la commande **SOUND**, tapez **1** à l'endroit requis pour le paramètre de l'<enveloppe de tonalité>. Nous expliquerons bientôt la création d'une enveloppe de tonalité.

Bruitage

C'est le dernier paramètre de la commande **SOUND**. Il peut varier de **1** à **31**. Ajoutez le paramètre **2** à la fin de la commande **SOUND** et écoutez l'effet obtenu. Faites-le passer à **27** et constatez la différence. Tapez :

```
10 SOUND 1,239,200,15,,,2
```

Remarquez que les deux paramètres précédant le paramètre de <bruitage> (,,,) ont été omis. Ils indiquent ainsi l'absence d'<enveloppes de volume> et de <tonalité>.

Création d'une enveloppe de volume

Elle s'obtient par la commande **ENV**. Sous sa forme de base, cette commande possède 4 paramètres. Elle se présente ainsi :

> **ENV** <numéro d'enveloppe>, <nombre de pas>, <amplitude de pas>, <durée du pas>

Comme précédemment, nous allons étudier ces paramètres un par un.

Numéro d'enveloppe

C'est un numéro de référence (entre **Ø** et **15**) appelable par la commande **SOUND**.

Nombre de pas

Ce paramètre spécifie le nombre de pas d'évolution du volume sonore entre le début du son et sa fin. Par exemple, pour une note de 10 secondes, vous pouvez obtenir 10 pas d'une seconde chacun. Dans ce cas-là, le paramètre du <nombre de pas> sera égal à **1Ø**.

Il peut varier de **Ø** à **127**.

Amplitude du pas

Chaque pas peut faire varier le volume de **Ø** à **15** par rapport au pas précédent. Les 16 volumes différents sont les mêmes que ceux de la commande **SOUND**. Cependant, le paramètre d'<amplitude du pas> peut varier de **-128** à **+127**, car le volume revient à **Ø** après avoir dépassé **15** (**16**, **32**, **48**, **64**, **8Ø**, **96**, etc.).

Durée du pas

Ce paramètre spécifie la durée de chaque pas par incrément de 0,01 seconde. Il varie de **Ø** à **255**, soit une durée maximale de 2,56 secondes (**Ø** est considéré comme **256**).

Notez que le <nombre de pas> multiplié par la <durée du pas> ne doit pas dépasser le paramètre de <durée> de la commande **SOUND**, car le son cesserait avant que le dernier pas n'ait été atteint (entraînant la perte du reste de l'enveloppe de volume).

De même, si le paramètre de <durée> de la commande **SOUND** dépasse le <nombre de pas> multiplié par la <durée du pas>, le son se poursuit après avoir atteint le dernier pas et garde le volume final.

Pour tester l'enveloppe de volume, tapez le programme :

```
10 ENV 1,10,1,100
20 SOUND 1,142,1000,1,1
run
```

La ligne **20** spécifie un son d'une période de **284** (LA international), durant 10 secondes avec un volume de départ de **1**, utilisant l'enveloppe de volume numéro **1** constituée de **10** pas, élevant chaque pas d'une unité de volume et durant une seconde (**100** x 0,01 sec).

Changez la ligne **10** de la manière suivante et faites run chaque fois pour entendre la différence :

```
10 env 1,100,1,10
10 env 1,100,2,10
10 env 1,100,4,10
10 env 1,50,20,20
10 env 1,50,2,20
10 env 1,50,15,30
```

Essayez finalement :

```
10 ENV 1,50,2,10
```

Vous pouvez constater que le niveau de son reste constant à mi-chemin. En effet, le nombre de pas (**50**) et l'écart entre les pas (0,1 seconde) donnent une variation du son de 5 secondes seulement, alors que la <durée> du son dans la commande **SOUND** (la ligne **20**) est de **1000**, soit 10 secondes.

Expérimentez les types de sons que vous pouvez créer.

Si vous le désirez, vous pouvez créer des enveloppes de volume plus complexes. Les 3 paramètres <nombre de pas>, <amplitude du pas> et <durée du pas> peuvent être repris jusqu'à quatre fois à la fin de la commande **ENV**, permettant de spécifier des portions d'enveloppe différentes.

Création d'une enveloppe de tonalité

La commande d'enveloppe de tonalité est **ENT**. Sous sa forme de base, elle a 4 paramètres. Elle se présente sous la forme :

> **ENT** <numéro d'enveloppe>, <nombre de pas>, <variation de période sonore affectée à chaque pas>, <durée du pas>

Regardons ces paramètres un par un.

Numéro d'enveloppe

C'est le numéro d'appel pour la commande **SOUND**. Il peut prendre une valeur de **0** à **15**.

Nombre de pas

Ce paramètre fixe le nombre de pas de la tonalité du son entre son début et sa fin. Si, par exemple, une note dure 10 secondes, vous pouvez obtenir **1Ø** pas d'une seconde chacun. Le paramètre <nombre de pas> est alors égal à **1Ø**.

Il peut varier de **Ø** à **239**.

Variation de période sonore affectée à chaque pas

Chaque pas peut faire varier la hauteur du son de **-128** à **+127** par rapport au pas précédent. Les pas négatifs augmentent la hauteur de la note. Les pas positifs l'abaissent. La période la plus courte correspond à **Ø**. Vous devez tenir compte de ces notions lorsque vous définissez une enveloppe de tonalité. Vous trouverez toutes les possibilités de périodes de ton au chapitre « Pour information... ».

Durée du pas

Ce paramètre spécifie la durée de chaque pas par incrément de 0,01 seconde. Il varie de **Ø** à **255**, soit une durée maximale de 2,56 secondes (**Ø** est considéré comme **256**).

Le <nombre de pas> multiplié par la <durée du pas> ne doit pas dépasser le paramètre de <durée> de la commande **SOUND**, car le son cesserait alors avant que le dernier pas n'ait été atteint (dans ce cas, le reste de l'enveloppe de tonalité est perdu).

De même, si le paramètre de <durée> de la commande **SOUND** dépasse le <nombre de pas> multiplié par la <durée du pas>, le son se poursuit après avoir atteint le dernier pas et garde la tonalité finale.

Pour tester l'enveloppe de tonalité, tapez le programme ci-dessous :

```
10 ENT 1,100,2,2
20 SOUND 1,142,200,15,,1
run
```

La ligne **20** spécifie un son ayant une période de **142** (LA international), durant 2 secondes avec un volume de départ de **15** (max), dépourvu d'enveloppe de volume (constatez l'omission du paramètre : **,,**) mais doté d'une enveloppe de tonalité N° **1**.

La ligne **1Ø** affecte au N° **1** une enveloppe de tonalité constituée de **1ØØ** pas, augmentant la période de **2** unités à chaque pas et ayant chacun une durée de 0,02 seconde (**2** x 0,01 s).

Modifiez maintenant la ligne **10** selon les trois manières indiquées et lancez le programme pour entendre l'effet obtenu.

```
10 ent 1,100,-2,2
10 ent 1,10,4,20
10 ent 1,10,-4,20
```

Remplacez la commande **SOUND** et l'enveloppe de tonalité en tapant :

```
10 ENT 1,2,17,70
20 SOUND 1,71,140,15,,1
30 GOTO 10
run
```

Appuyez deux fois sur **[ESC]** pour interrompre le programme.

Vous pouvez maintenant combiner l'enveloppe de volume, l'enveloppe de tonalité et la commande **SOUND** pour obtenir des sons différents. Commencez par taper :

```
10 ENV 1,100,1,3
20 ENT 1,100,5,3
30 SOUND 1,142,300,1,1,1
run
```

Puis remplacez la ligne **20** par :

```
20 ENT 1,100,-2,2
run
```

Remplacez maintenant toutes les lignes par :

```
10 ENV 1,100,2,2
20 ENT 1,100,-2,2
30 SOUND 1,142,200,1,1,1
run
```

Si vous désirez créer des enveloppes de tonalité plus complexes, vous pouvez répéter jusqu'à quatre fois les trois paramètres : <nombre de pas>, <variation de période sonore affectée à chaque pas> et <durée du pas> à la fin de la commande **ENT** afin de spécifier des portions différentes pour une même enveloppe. Essayez plusieurs versions. Ajoutez à la commande **SOUND** le paramètre permettant d'ajouter du bruitage et des portions supplémentaires d'enveloppe de tonalité et de volume.

Le chapitre « Liste complète des mots clés du BASIC Schneider 6128 » décrit en détail les commandes de son. Si vous désirez découvrir les possibilités sonores de votre ordinateur, reportez-vous au paragraphe « Sons et musique » du chapitre « A vos heures de loisir... ».

Partie 10 : Introduction à AMSDOS et CP/M...

Qu'est-ce qu'AMSDOS ?

A la mise sous tension ou lors d'une réinitialisation, le système se place par défaut sous AMSDOS, abréviation d'AMstrad Disc Operating System (système d'exploitation de disque Schneider), offrant les commandes de traitement de fichiers et les fonctions :

```
LOAD "nomfich"
RUN "nomfich"
SAVE "nomfich"
CHAIN "nomfich"
MERGE "nomfich"
CHAIN MERGE "nomfich"
OPENIN "nomfich"
OPENOUT "nomfich"
CLOSEIN
CLOSEOUT
CAT
EOF
INPUT #9
LINE INPUT #9
LIST #9
PRINT #9
WRITE #9
```

AMSDOS fournit en outre plusieurs commandes supplémentaires de gestion de disquettes.

Ces commandes, appelées commandes externes, sont précédées du symbole | (vous entrez cette barre | en appuyant simultanément sur les touches **[SHIFT]** et @)

Les commandes externes les plus fréquentes sont :

```
|a
|b
```

| tape (qui peut être subdivisée en | **tape.in** et en | **tape.out**)
| disc (qui peut être subdivisée en | **disc.in** et en | **disc.out**)

Les commandes | **a** et | **b** permettent aux systèmes à deux unités de disquettes d'indiquer à l'ordinateur l'unité vers laquelle il doit diriger les commandes ultérieures.

Si vous tapez, par exemple :

```
|a
load "nomfich"
```

...vous indiquez à l'ordinateur qu'il doit charger en mémoire le programme spécifié, se trouvant sur une disquette placée dans l'unité A.

Si vous ne spécifiez pas | **a** ou | **b** lors de la réinitialisation, le système exploite par défaut l'unité A. Si vous n'utilisez que l'unité de disquette intégrée au système, elle est alors considérée comme l'unité A et vous n'avez donc aucun besoin de spécifier | **a** ou | **b**. Avec un système à une seule unité, un | **b** fait apparaître le message d'erreur suivant :

```
Drive B: disc missing
Retry, Ignore or Cancel
```

...auquel vous devez répondre **C** (pour cancel - annuler).

Que faire pour utiliser une cassette ?

La commande | **tape** indique à l'ordinateur qu'il doit gérer le chargement, la sauvegarde, etc., vers une unité de cassette externe et non vers la disquette. Si | **tape** n'est pas entré, l'ordinateur, à chaque mise en route ou reset, dirige toutes les opérations automatiquement vers la disquette. Pour revenir sur disquette après un | **tape**, vous devez rentrer la commande : | **disc**. Par contre, vous pouvez avoir besoin de charger un programme sur cassette et de sauvegarder des données sur disquette. Faites alors appel à la commande :

```
|tape.in
```

Cette commande dit à l'ordinateur : tu dois lire sur la cassette et écrire sur la disquette.

Inversement, pour lire le disque et écrire sur la cassette, vous devez entrer la commande | **disc.in** pour annuler le | **tape.in** entré précédemment, puis | **tape.out** pour diriger toutes sorties de données vers la cassette.

On peut ainsi avoir des | **tape.in** et des | **tape.out** qui vont annuler des | **disc.in** et des | **disc.out**, et vice-versa.

Vous trouverez des détails plus précis concernant les entrées et sorties aux chapitres « Utilisation des disquettes et des cassettes » et « Les éléments de l'AMSDOS et du CP/M » (chapitres 4 et 5).

Duplication de programmes d'une disquette à l'autre ou sur cassette

La partie 7 du Cours Elémentaire décrit le formatage d'une disquette vierge à l'aide du programme **DISCKIT 3** sur la face 1 de l'une des disquettes système CP/M et la duplication des programmes d'une disquette à l'autre. Considérant les informations ci-dessus les commandes :

```
|tape |disc |tape.in |tape.out |disc.in |disc.out  |a |b
```

...vous permettront de charger et de sauvegarder vos données sur cassette (si une unité de cassette est connectée à l'ordinateur) ou disquette, dans l'unité A ou B.

Les commandes externes :

```
|dir  |drive  |era  |ren  |user
```

...sont traitées dans le chapitre 5 « Les éléments de l'AMSDOS et du CP/M ».

Duplication/copie intégrale d'une disquette

Vous pouvez, à l'aide du programme **DISCKIT3** (face 1 d'une des disquettes système CP/M), copier intégralement une disquette sur une autre.

Cette méthode vous servira, entre autres, à faire des copies de sauvegarde de vos disquettes système.

Insérez l'une des disquettes système dans l'unité, face 1 vers le haut, et tapez :

```
|cpm
```

A la suite du message **A** >, tapez :

```
disckit3
```

Au bout de quelques secondes, le message initial du DISC KIT apparaît en haut de l'écran, suivi de :

```
One drive found
```

Ce message indique que vous êtes sous l'utilitaire DISC KIT et que l'ordinateur a détecté l'exploitation d'une seule unité de disquette (celle qui est intégrée à l'ordinateur).

Si une unité supplémentaire est reliée au système, l'affichage indique :

```
Two drives found
```

et ce menu apparaît au bas de l'écran :

Copy	7	
Format	4	
Verify	1	
Exit from program	Ø	

Il s'agit du menu principal du DISC KIT. Les numéros inscrits dans les cases correspondent aux touches de fonction situées à droite du clavier (**fO**, **f1**, **f3** et **f7**). Faites votre sélection à l'aide de la touche voulue.

Appuyer sur la touche de fonction **fO**, à ce stade des opérations, vous fait sortir du programme DISC KIT pour vous ramener sous le système CP/M (message **A** >).

Puisque nous voulons copier une disquette, appuyez sur **f7**.

ATTENTION

LA COPIE SUR UNE DISQUETTE DEJA ENREGISTREE ENTRAINERA L'EFFACEMENT DE SON CONTENU.

Copie sur un système à une seule unité

Si vous utilisez un système à une seule unité (pas d'unité supplémentaire raccordée), vous recevez à l'écran le message :



Vous devez alors retirer la disquette système de l'unité et insérer la disquette à dupliquer. Pour dupliquer la disquette système, il suffit de la laisser dans l'unité.

Si la disquette source (celle dont vous souhaitez copier le contenu) se trouve dans l'unité, tapez **Y**.

L'ordinateur examine le format de la disquette et l'affiche en haut de l'écran, pour votre information.

Au bout d'un moment, ce message vous est adressé :

```
Insert disc to WRITE
Press any key to continue
```

Vous devez alors retirer la disquette source de l'unité, insérer la disquette cible (celle sur laquelle vous désirez copier les données), puis appuyer sur une touche quelconque du clavier.

Les informations sur le format de la disquette cible sont affichées en haut de l'écran, même s'il s'agit d'une disquette vierge non formatée.

Si la disquette cible n'est pas correctement formatée (voire pas formatée du tout), le formatage s'effectuera pendant la duplication et vous verrez apparaître le message :

```
Disc isn't formatted (or fautly)
Going to format while copying
Disc will be system format
```

... ou un message similaire, suivant les disquettes cible et source.

Lorsque l'ordinateur est prêt pour la lecture de la disquette source, il vous communique ce message :

```
Insert disc to READ
Press any key to continue
```

... vous devez alors réinsérer la disquette source.

L'achèvement de l'opération de copie après répétitions de cette manipulation des disquettes source et cible, est signalé par le message :

```
Copy completed
Remove disc
Press any key to continue
```

... suivez alors les instructions affichées à l'écran, qui vous laissent le choix entre copier une autre disquette (en tapant **Y**) et revenir au menu principal du DISC KIT.

Protection en écriture

Vous ne pouvez pas effectuer de copie sur une disquette dont le trou de protection en écriture est ouvert. Si vous tentez de le faire, vous obtenez ce message :

```
Disc write-protected
Insert disc to WRITE
R-etry or C-ancel
```

Tapez alors **C** pour annuler, retirez la disquette et insérez-en une autre appropriée à la copie.

Veillez à ne jamais fermer le trou de protection en écriture des disquettes dont vous souhaitez conserver le contenu. C'est NOTAMMENT le cas des disquettes système CP/M.

Duplication/copie dans un système à deux unités

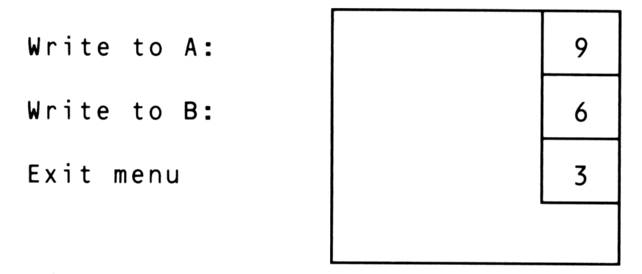
Suivez les instructions ci-dessus, chargez le programme **DISCKIT3** situé sur la face 1 d'une des disquettes système et sélectionnez l'option **Copy** du menu principal du DISC KIT au moyen de la touche **f7**.

Vous verrez alors apparaître un nouveau menu :

Read from A:	8
Read from B:	5
Exit menu	2

Lors de la duplication de disquettes dans un système à deux unités, insérer et retirer les disquettes cible et source n'est plus nécessaire. Le menu ci-dessus vous permet de sélectionner l'unité dans laquelle vous désirez insérer la disquette source. Appuyez sur **f8** pour copier à partir de la disquette de l'unité A.

Un troisième menu apparaît alors à l'écran :



Ce menu vous permet de sélectionner l'unité dans laquelle se trouvera la disquette cible. Rien ne vous empêche de choisir la même unité pour les disquettes source et cible, mais vous serez alors obligé d'intervertir les disquettes, négligeant les avantages d'un système à deux unités.

Pour plus de rapidité et de commodité, appuyez sur la touche **f6** pour effectuer la copie sur une disquette insérée dans l'unité B.

Il ne vous reste plus qu'à placer les disquettes dans les unités appropriées et appuyer sur Y pour lancer la duplication.

Cette fois encore, les informations sur le format des deux disquettes vont s'afficher et si la disquette cible est mal formatée ou pas formatée du tout, cette lacune sera comblée lors de la copie.

La fin de la duplication est signalée par :

```
Copy completed
Remove both discs
Press any key to continue
```

Vous devez alors retirer les DEUX disquettes pour poursuivre. Vous pouvez refaire une duplication en tapant **Y**, ou revenir au menu principal du DISC KIT à partir de toute autre touche.

Vérification des disquettes

Le programme **DISCKIT3** vous offre également la possibilité de vérifier les disquettes.

Le système affiche les informations concernant le format de la disquette et effectue la lecture de tous les fichiers qu'elle contient. Toute erreur détectée à l'intérieur d'un fichier est signalée à l'écran.

Pour vérifier une disquette, insérez l'une des disquettes CP/M, face 1 vers le haut, et tapez :

```
|cpm
```

A la suite du message **A**, tapez :

```
disckit3
```

Sélectionnez l'option **Verify** du menu principal du DISC KIT (touche **f1**) et suivez les instructions affichées.

Si vous opérez sur un système à deux unités, vous pouvez vérifier une disquette insérée dans les unités A ou B.

Une fois la disquette à vérifier dans l'unité, tapez **Y** pour lancer l'opération.

La fin de la vérification est signalée par:

```
Verify completed
Remove disc
Press any key to continue
```

Si vous opérez sur un système à deux unités, vous devez retirer les DEUX disquettes avant de pouvoir poursuivre. Effectuez alors une autre vérification en tapant **Y**, ou revenez au menu principal du DISC KIT en appuyant sur toute autre touche.

Utilisation du programme DISC KIT sous CP/M2.2

Pour formater, copier ou vérifier des disquettes créées avec ou pour les systèmes CPC664 ou CPC464 + DDI1, donc sous CP/M2.2, vous disposez d'un programme DISC KIT, situé sur la face 4 d'une des disquettes système. Ce programme s'appelle **DISCKIT2** et fonctionne de la même manière que le **DISCKIT3** déjà décrit.

Pour lancer le **DISCKIT2**, insérez la disquette adéquate, face 4 vers le haut, et tapez:

```
|cpm
```

A l'apparition du message initial du CP/M2.2 et de **A**, tapez:

```
disckit2
```

Il vous suffit ensuite de sélectionner les options requises et suivre les instructions à l'écran de la même façon qu'avec le **DISCKIT3**.

Remarque: Pour la duplication, le programme **DISCKIT2** utilise la mémoire de l'écran. Vous verrez donc apparaître à l'écran, pendant cette procédure, une image aléatoire.

Complément d'information

Pour de plus amples informations sur l'utilisation des programmes des disquettes système CP/M, consultez les chapitres 4 à 9.

En dernier ressort, vérifiez que vous avez bien respecté les indications donnés au début du manuel, sous la rubrique « IMPORTANT » :

REMARQUES RELATIVES A L'INSTALLATION : 5, 6, 7.
REMARQUES RELATIVES AU FONCTIONNEMENT: 1, 2, 3, 4, 5, 6, 7, 9.

Partie 11 : Introduction à BANK MANAGER

Utiliser les autres 64Ko de la mémoire

Le CPC6128 possède une RAM (Random Access Memory ou « mémoire vive ») de 128Ko, divisée en deux parties de 64Ko. CP/M Plus utilise en permanence la totalité de ces 128Ko, alors que le BASIC n'exploite que la première partie de 64Ko disponible. Il serait dommage de laisser à l'abondon ces 64Ko au cours d'une programmation en BASIC. C'est pourquoi un programme a été mis à disposition pour l'exploitation de cet espace mémoire inutilisé. Ce programme offre des commandes spéciales supplémentaires pour permettre l'utilisation de la seconde partie de RAM, soit pour le stockage des images d'écran, soit pour la mémorisation de chaînes.

Ce programme fournissant des commandes supplémentaires s'intitule BANK MANAGER, « gestionnaire de bloc », le terme technique « bloc » désignant une partie de la mémoire.

Utiliser BANK MANAGER pour les images d'écran

Le 6128 affiche en permanence une image à l'écran. Il exploite pour ce faire 16Ko de mémoire pour stocker les informations concernant la couleur et la brillance de chaque pixel de l'écran. Le 6128 gère six blocs de 16Ko, ce qui signifie six images d'écran stockées simultanément dans la mémoire de l'ordinateur. BANK MANAGER vous permet de jongler et d'afficher un maximum de cinq images d'écran avec le BASIC.

A la mise sous tension, le système d'affichage exploite un bloc mémoire de 16Ko; nous l'appellerons « bloc 1 ». Les quatre autres écrans sont contenus dans la seconde partie de 64Ko, dans les blocs 2, 3, 4 et 5.

Seul le bloc 1 (contenu dans la première partie de la mémoire) peut servir à l'affichage d'un écran. Pour visualiser un bloc stocké dans la deuxième partie (blocs 2 à 5), il faudra donc le placer dans le bloc 1. BANK MANAGER fournit toutes les commandes nécessaires pour déplacer les écrans: **|SCREENCOPY**, par exemple, transfère un écran sur un autre écran en écrasant le contenu de ce dernier. **|SCREENSWAP**, pour sa part, permute le contenu des écrans.

Tout comme AMSDOS évoqué précédemment, BANK MANAGER dispose de « commandes externes » commençant par le symbole | (barre verticale), que vous obtenez en appuyant simultanément sur **[SHIFT]** et @.

Utilisation de BANK MANAGER

Réinitialisez l'ordinateur en appuyant sur les touches **[CONTROL] [SHIFT]** et **[ESC]**, insérez l'une des disquettes système, face 1 vers le haut, puis tapez :

```
RUN "BANKMAN"
```

La procédure de chargement de ce programme est détaillée au chapitre 7, partie 13, concernant les extensions résidentes du système (RSX). Une certaine compréhension de ces extensions et de la réservation d'espace mémoire est recommandée pour l'utilisation de ces sous programmes. Quoi qu'il en soit, pour les exemples qui suivent, vous n'avez pas à maîtriser la procédure de chargement.

Tapez :

```
MODE 1
PRINT "CECI EST L'ECRAN PAR DEFAUT"
|SCREENSWAP,1,2
```

Le texte doit alors disparaître de l'écran. Vous cherchez maintenant ce qu'était le contenu de l'écran 2 (bloc 2 de la mémoire). Si l'ordinateur vient d'être mis sous tension, vous avez probablement à l'écran une image aléatoire. Pour l'effacer, tapez :

```
MODE 1
```

puis :

```
PRINT "CECI EST L'ECRAN 2"
|SCREENSWAP,1,2
```

Le texte initial s'affiche de nouveau. Si vous répétez la commande **|SCREENSWAP,1,2**, vous constaterez que les contenus des deux écrans ont été intervertis. Vous pouvez ainsi échanger les contenus des cinq écrans disponibles, mais n'oubliez pas que seuls les échanges impliquant l'écran 1 seront visibles à l'écran.

L'autre commande disponible est **|SCREENCOPY**. Elle vous permet de copier un écran sur un autre avec écrasement des anciennes données.

Tapez :

```
MODE 1
PRINT "CECI EST L'ECRAN A COPIER"
|SCREENCOPY,2,1
```

Le contenu de l'écran 1 est copié sur l'écran 2. Si vous inversez les paramètres en tapant :

```
MODE 1

|SCREENCOPY,1,2
```

le contenu de l'écran affiché est écrasé par celui de l'écran 2.

Le premier paramètre est donc l'écran sur lequel s'effectue la copie, et le second paramètre, l'écran à partir duquel s'effectue celle-ci.

Lorsque vous copiez des écrans et que les **MODE**s des écrans sont différents, vous obtenez un affichage disjoint, de même si vous avez effectué un défilement d'écran depuis la dernière commande **MODE**. Les commandes d'écran de BANK MANAGER sont plutôt destinées aux écrans graphiques qu'aux écrans de texte car les défilements y sont moins fréquents.

Utiliser BANK MANAGER pour le stockage de chaînes

BANK MANAGER dispose de quatre commandes supplémentaires destinées à l'utilisation des 64Ko de mémoire supplémentaire pour archivage de chaînes alphanumériques.

La plupart des programmes peuvent se scinder en deux parties: d'une part, les instructions proprement dites et d'autre part, les données du programme. Un programme de gestion de base de données comme un carnet d'adresse en est un bon exemple. Un tel programme utilise un tableau de chaînes pour le stockage des noms et des adresses correspondant aux différentes personnes répertoriées.

Les chaînes sont stockées dans la seconde partie de 64Ko, l'une après l'autre et bout à bout. L'espace de stockage de ces chaînes peut se diviser en compartiments, appelés enregistrements. La longueur fixe d'un enregistrement est comprise entre 2 et 255 caractères et la longueur d'une chaîne en BASIC varie selon son contenu. Les enregistrements servent à fournir des compartiments de taille standard propres au rangement des chaînes d'information sous une forme ordonnée. A chaque stockage ou recherche de données dans un enregistrement, le système passe automatiquement à l'enregistrement suivant pour une nouvelle opération. Cet enregistrement, appelé « enregistrement courant », est automatiquement utilisé, sauf spécification contraire.

Ce système de gestion de mémoire est nommé « RAMdisc », car il fonctionne comme un disque à accès aléatoire. Ici, le support est remplacé par la RAM.

Lisez les descriptions des différentes commandes pour comprendre leur raison d'être, sinon leur fonctionnement, et passez aux exemples.

La première commande RAMdisc est |**BANKOPEN**. Elle spécifie le nombre de caractères contenus dans chaque enregistrement. Voici sa syntaxe :

```
|BANKOPEN,n
```

où n est le nombre de caractères de l'enregistrement, compris entre 0 et 255 ; mais les valeurs 0 et 1 provoquent des effets inattendus.

BANKWRITE stocke une chaîne dans l'enregistrement en cours; le pointeur d'enregistrement est incrémenté pour pointer sur l'enregistrement suivant, prêt pour la prochaine opération. Voici sa syntaxe :

```
|BANKWRITE,@r%,a$
```

ou

```
|BANKWRITE,@r%,a$,n
```

où **r%** est une variable entière contenant un code de retour pour information sur le déroulement de l'opération, et **a$** une variable alphanumérique contenant les caractères à écrire dans l'enregistrement. Dans le premier exemple, l'enregistrement est l'enregistrement en cours. Dans le second, le paramètre n, facultatif, spécifie l'enregistrement sur lequel s'effectuera l'opération d'écriture.

BANKREAD lit l'enregistrement en cours et renvoie son contenu dans une chaîne. Le pointeur d'enregistrement est ensuite incrémenté pour pointer sur l'enregistrement suivant, prêt pour la prochaine opération. La lecture d'un enregistrement le laisse inchangé: il peut donc être lu et relu indéfiniment. Voici sa syntaxe:

```
|BANKREAD,@r%,a$
```

ou

```
|BANKREAD,@r%,a$,n
```

où **r%** est une variable entière contenant un code de retour pour information sur le déroulement de l'opération, et **a$** une variable alphanumérique dans laquelle le contenu de l'enregistrement est transféré. Dans le premier exemple, l'enregistrement est l'enregistrement en cours. Dans le second, le paramètre n, facultatif, spécifie l'enregistrement à lire.

La dernière commande, |**BANKFIND**, recherche une chaîne particulière dans l'ensemble des enregistrements. Si la chaîne est retrouvée, la commande envoie le numéro de l'enregistrement. Voici sa syntaxe:

```
|BANKFIND,@r%,a$
```

ou

```
|BANKFIND,@r%,a$,n
```

ou

```
|BANKFIND,@r%,a$,n,m
```

où **r%** est une variable entière qui renvoie soit le numéro de l'enregistrement dans lequel la chaîne a été trouvée, soit un code indiquant que la chaîne n'a pas été trouvée; **a$** est la chaîne à rechercher. Le paramètre facultatif n spécifie l'enregistrement du début de la recherche; s'il n'est pas spécifié, la recherche démarre à l'enregistrement en cours. Le deuxième paramètre facultatif, m, spécifie l'enregistrement de fin de recherche même si la chaîne n'est pas retrouvée. En cas d'omission de m, la recherche se poursuit, jusqu'à concurrence des 64Ko de mémoire, pouvant dans certains cas continuer au-delà du dernier enregistrement écrit.

Passons maintenant à un exercice. Si vous avez déjà lancé le programme BANK MANAGER pour prendre connaissance des commandes d'échange d'écrans et que vous n'avez pas réinitialisé l'ordinateur, ces commandes supplémentaires sont désormais résidentes. Dans le cas contraire, vous devez insérer l'une des disquettes système, face 1 vers le haut, et taper :

```
RUN "BANKMAN"
```

puis :

```
|BANKOPEN,20
```

La longueur de l'enregistrement est ainsi fixée à 20 caractères et l'enregistrement en cours est 0.Tapez alors :

```
a$="PREMIERE ENTREE"+SPACE$(5)
```

La chaîne **a$** comprendra exactement 20 caractères.

Tapez maintenant :

```
r%=0
```

afin d'initialiser la variable **r%**,

puis :

```
|BANKWRITE,@r%,a$
```

a$ va donc s'écrire dans le premier enregistrement (0). Tapez ensuite :

```
d$=SPACE$(20)
|BANKREAD,@r%,d$,0
PRINT d$
```

La première commande initialise **d** avec 20 espaces, suffisants pour contenir l'enregistrement en cours de lecture. La deuxième commande lit l'enregistrement 0 et place le résultat dans **d$**. La commande **|BANKWRITE** incrémente le pointeur d'enregistrement d'une unité: il faut donc spécifier l'enregistrement 0 pour la lecture. En effet, souvenez-vous que la lecture commence par défaut à l'enregistrement en cours. Le résultat de la lecture est finalement affiché, **d$** doit donc contenir **« PREMIERE ENTREE »** et 5 espaces.

Tapez maintenant :

```
b$="DEUX"+SPACE$(16)
c$="TROIS"+SPACE$(15)
|BANKWRITE,@r%,b$,1
|BANKWRITE,@r%,c$
```

Ces instructions placent **b$** et **c$** dans les enregistrements 1 et 2. Le paramètre facultatif de la première commande |**BANKWRITE** permet de placer **b$** dans l'enregistrement 1. L'enregistrement en cours passant ensuite à l'enregistrement suivant, il n'est donc pas nécessaire de spécifier l'enregistrement de destination de **c$**, qui sera placé automatiquement dans l'enregistrement 2.

Tapez :

```
PRINT r%
```

Le résultat de **PRINT r%** dans cet exemple doit être 2. On peut considérer ce numéro comme le dernier enregistrement sur lequel une opération a été effectuée ou bien encore comme l'enregistrement courant diminué d'une unité. Dans notre exemple, le dernier enregistrement d'écriture est 2 et l'enregistrement suivant sera 3.

L'intérêt de ce « code de retour » est de nous renseigner sur le déroulement de l'opération qui vient de s'effectueur. Lorsque celle-ci a abouti, le code est positif et correspond à un numéro d'enregistrement. Sinon, le code est négatif et renvoie à un code d'erreur. Les commandes |**BANKWRITE** et |**BANKREAD** peuvent engendrer deux codes d'erreur :

1 - fin de fichier, indique que tous les enregistrements ont été passés en revue ou que l'enregistrement spécifié n'existe pas ;

2 - erreur lors d'un changement de bloc, ce qui ne devrait jamais se produire.

Passons à d'autres exemples :

```
d$=STRING$(20,"X")
|BANKOPEN,20
FOR n=1 to 3:|BANKWRITE,@r%,d$:next
```

d$ contiendra 20 **x**. |**BANKOPEN** réinitialise le pointeur d'enregistrement à 0, ce qui provoque l'écrasement des enregistrements 0, 1 et 2 par le contenu de **d$** lors de l'exécution de la commande |**BANKWRITE**.

Tapez maintenant :

```
a$="PREMIER"
|BANKWRITE,@r%,a$,0
```

Cette instruction écrit « **PREMIER** » dans l'enregistrement0, écrasant ainsi quelques **X**. A présent, réinitialisez **d$** avec 20 espaces :

```
d$=SPACE$(20)
```

Tapez :

```
|BANKREAD,@r%,d$,0
```

Le résultat de la lecture de l'enregistrement **0** est placé dans la variable **d$**. Récapitulons :

Les trois enregistrements contiennent des **X**. Dans l'enregistrement 0, nous avons introduit le mot « **PREMIER** ». Nous avons rempli **d$** d'espaces, puis introduit dans **d$** le résultat de la lecture de l'enregistrement 0. Tapez maintenant :

```
PRINT d$
```

Le résultat doit être« **PREMIERXXXXXXXXXXXXX** ». Ceci illustre une particularité importante de ces commandes. Si la chaîne placée dans un enregistrement ne le remplit pas entièrement, les caractères non écrasés demeurent dans cet enregistrement. Il est donc préférable d'y introduire une chaîne d'espaces (**CHR$(32)**) avant d'y stocker de nouvelles données. Ceci évite, lors de la relecture d'une chaîne, de trouver des caractères qui ne devraient pas y être. Cette constatation concerne également la chaîne où s'inscrit le résultat d'une lecture d'enregistrement. Si la chaîne est plus longue que l'enregistrement, certains caractères en fin de chaîne resteront inchangés, ce qui explique la réinitialisation de **d$** (par remplissage d'espaces) avant qu'y soit placé le résultat de la lecture de l'enregistrement 0.

Il est possible d'écrire une chaîne dans un enregistrement contenant moins de caractères que celle-ci. Dans ce cas, les caractères excédants seront ignorés.

De même, il est possible de lire le contenu d'un enregistrement dans une chaîne comportant moins de caractères que celui-ci. Là encore, les caractères excédants (lus dans l'enregistrement) seront ignorés. Habituellement, le BASIC augmente automatiquement la longueur de la chaîne pour qu'elle accepte les caractères excédentaires. Ce n'est pas le cas avec les commandes externes.

Pour terminer, voyons la commande |**BANKFIND**. Elle permet de rechercher une chaîne parmi un groupe d'enregistrements. Par exemple, si l'enregistrement 24 commence par « **FRED** », cette commande permettra de le trouver :

```
|BANKFIND,@r%,"FRED"
```

Cette commande est extrêmement utile pour les programmes de gestion de bases de données, pour la recherche d'un nom ou d'une adresse.

BANKFIND commence sa recherche à l'enregistrement courant et pousuit jusqu'à trouver la chaîne en question, ou la fin de la seconde partie de 64Ko de mémoire où sont stockés les enregistrements.

Il est possible de spécifier l'enregistrement de départ de la recherche en ajoutant un numéro d'enregistrement à la fin de la commande.

Il est également possible d'introduire un autre numéro d'enregistrement à la suite du premier pour indiquer l'enregistrement de fin de recherche.

BANKFIND peut aussi servir à retrouver une chaîne situé ailleurs qu'au début d'un enregistrement. Pour ce faire, on placera devant les caractères recherchés des **CHR$(0)** qui, traités comme des jokers (tel **« ? »** dans les noms de fichiers sous CP/M), pourront ainsi remplacer n'importe quel caractère. Voici un exemple :

```
a$=STRING$(10,0)+"FRED"
|BANKFIND,@r%,a$,0
```

Et voici retrouvé le premier enregistrement concernant le mot **« FRED »** situé entre le onzième et quatorzième caractères. Les dix premiers caractères pourront contenir un numéro de téléphone ou tout autre information, que la commande |**BANKFIND** ignorera.

Le numéro de l'enregistrement où se trouve la chaîne sera placé dans la variable entière r% servant de code de retour (si **« FRED »** a été trouvé), sinon, le code de retour sera **-3**.

Complément d'information

Pour plus de détails sur BANK MANAGER, reportez-vous au chapitre 8 et aux parties 13 et 14 du chapitre 7 concernant les extensions résidentes du système (RSX).

Enfin, assurez-vous que vous avez bien respecté les avertissements donnés au début de ce manuel, sous la rubrique « IMPORTANT » :

REMARQUES RELATIVES A L'INSTALLATION : 1, 2, 3, 9.

Voici la fin de la partie 10 du Cours Elémentaire sur le CPC664. Vous connaissez désormais la plupart des touches du clavier, les commandes BASIC les plus simples, le formatage d'une disquette vierge pour sa préparation à l'emploi et l'exécution des fonctions de base sur disquette comme : **LOAD**, **SAVE**, **CAT**, etc... ainsi que quelques commandes élémentaires AMSDOS et CP/M. La partie suivante approfondit certains aspects de l'informatique et du BASIC Schneider. Il détaille l'unité de disquette du système, présente AMSDOS et CP/M et vous initie au nouveau langage de Digital Research : Dr.LOGO.

Bonne chance... et bonne lecture !

Chapitre 2
Passons aux choses sérieuses...

Après la lecture du Cours Elémentaire, vous savez mettre votre ordinateur en marche, lui faire répéter plusieurs fois une même opération en utilisant la boucle **FOR NEXT** et lui demander de vérifier si (**IF**) une condition est remplie pour qu'il exécute alors (**THEN**) une instruction.

Mais vous devez en avoir assez de voir votre nom s'afficher sur tout l'écran et vous voudriez maintenant commencer à faire vraiment de l'informatique, utile ou amusante. Vous trouverez dans le chapitre suivant la liste des mots clés du BASIC Schneider, accompagnés de tout ce qu'il faut savoir sur leur « syntaxe » et leur fonction. Lorsque vous la possèderez à fond, votre ordinateur et vous ne connaîtrez d'autres limites que celles de votre imagination.

Si c'est la première fois que vous vous servez d'un ordinateur, vous ressentirez peut-être quelque appréhension à l'idée de « faire de la programmation ». Rassurez-vous : c'est bien plus simple que vous ne l'imaginez et certainement moins compliqué que les spécialistes ne voudraient le faire croire avec leur jargon ! Dites-vous que BASIC n'est pas une nouvelle langue à apprendre, mais plutôt un dérivé de l'anglais auquel, pour gagner du temps, on a ajouté quelques abréviations. **CLS** n'est donc pas une formule magique, mais la forme abrégée de **CL**ear **S**creen (Vide l'Ecran).

Finie la peur du BASIC, vous serez bientôt surpris de voir comme il est agréable, après quelques heures de programmation, de récolter les premiers fruits de ses efforts. La programmation est un exercice passionnant, surtout pour un débutant qui aborde un langage et une machine pour la première fois. Souvenez-vous bien que, tant que vous n'écrivez pas par inadvertance sur la disquette du système d'exploitation CP/M, rien de ce que vous entrez au clavier ne peut endommager votre ordinateur. Aussi, n'hésitez pas à innover !

Mais par où commencer ?

C'est souvent le début d'un programme qui pose le plus de problèmes au néophyte. Evitez surtout de vous jeter sur le clavier avant d'avoir réfléchi à ce que vous voulez faire.

Vous devez d'abord chercher à établir, d'une part, ce que le programme est censé faire et, d'autre part, la manière dont les résultats vous seront présentés, c'est-à-dire ce qui apparaîtra à l'écran lors de l'exécution du programme.

Une fois ceci établi, vous pouvez entamer la rédaction proprement dite du programme, en songeant constamment qu'il doit se dérouler sans à-coups du début à la fin, avec un minimum de sauts (**GOTO**) en tous sens. Un programme bien écrit doit se lire aisément : songez au casse-tête qui vous attend lors de sa mise au point, ou de son « débogage », comme on dit en jargon informatique.

Soyez tranquille, BASIC est un langage extrêmement tolérant : il vous remettra souvent dans le droit chemin d'un simple message d'erreur affiché à l'écran. Si une idée vous vient après coup, il ne fera aucune difficulté pour insérer une nouvelle ligne entre deux lignes existantes.

Ecrivons un petit programme

Et maintenant, au travail ! Nous allons écrire un programme permettant d'enregistrer le nom et le numéro de téléphone de vos amis. Ce sera le programme « Répertoire Téléphonique ». Posons-nous d'abord les deux questions primordiales: « Que doit-il faire ? », et « Comment les résultats seront-ils présentés ? »

Supposons que nous voulions un programme permettant d'enregistrer 100 noms et numéros de téléphone. Il suffira d'entrer au clavier le nom d'une personne donnée pour obtenir aussitôt son numéro de téléphone. Ce programme devra de plus être capable d'afficher la liste de toutes les informations mémorisées, au cas où nous aurions oublié sous quelle forme un nom a été enregistré. Vous remarquez que nous sommes déjà en train de penser à la présentation des résultats.

Très bien, à vos claviers ! Commençons par le titre :

```
10 REM repertoire telephonique
```

Donner un titre n'est pas une obligation. Il vous aidera pourtant beaucoup à vous y retrouver lorsque vous aurez accumulé une quantité de programmes.

Nous voulons donc entrer (**INPUT**) une chaîne de caractères (un nom) dans une variable. Cette variable, nous l'appellerons **NOM$**. De même, nous aurons une variable pour les numéros de téléphone, que nous appellerons **TEL$**.

Vous vous souvenez que, dans les exemples de programmes du Cours Elémentaire, l'instruction **INPUT** vous avait permis d'affecter une valeur à une variable. Ainsi, avec les lignes :

```
20 INPUT "entrez le nom";NOM$
30 INPUT "entrez le numero de telephone";TEL$
run
```

...nous pourrons entrer un nom (disons « Paul »), puis un numéro de téléphone (par exemple 1 206 66 60).

Ces informations sont maintenant stockées par le programme, mais nous n'obtenons encore aucun résultat à l'écran. Il nous faut donc écrire une partie de programme permettant de retrouver et d'afficher ces informations. Pour obtenir la valeur actuelle de **NOM$** et **TEL$**, on aura recours aux instructions :

```
PRINT NOM$

... et...

PRINT TEL$
```

Mais attendez ! Nous avions bien dit qu'avec notre programme nous voulions enregistrer jusqu'à 100 noms et numéros de téléphone ? Nous n'allons tout de même pas écrire 100 instructions **INPUT** avec un nom de variable différent pour chacune, puis 100 instructions **PRINT** pour afficher la liste à l'écran !!! Non, bien sûr : l'ordinateur résout ce problème au moyen d'un instrument appelé « tableau ». Le tableau permet d'affecter à une même variable la « dimension » (le nombre de valeurs) désirée (100 dans notre cas). Pour connaître l'une de ces valeurs, il suffit d'entrer le nom de la variable suivi (entre parenthèses) du numéro de référence correspondant. Ce numéro s'appelle un « indice » et une expression telle que **NOM$(27)** s'appelle une « variable indicée ». Nous sommes donc en mesure, en utilisant une variable numérique x, de travailler sur l'ensemble des valeurs **NOM$(x)** de notre tableau : il suffit de faire prendre à **x** des valeurs de 1 à 100 dans une boucle **FOR NEXT** (**FOR x=1 TO 1ØØ**). Chaque fois que **x** est augmenté de 1, l'indice est également modifié et désigne l'élément suivant du tableau, un nom dans ce cas.

Nous avons besoin, pour les variables **NOM$** et **TEL$**, de deux tableaux de dimension 100. Avant d'utiliser un tableau, il faut avoir déclaré sa **DIM**ension. Remplaçons donc nos lignes **2Ø** et **3Ø** par ces déclarations.

```
20 DIM NOM$(100)
30 DIM TEL$(100)
```

Nos variables ainsi définies, écrivons un programme permettant d'introduire les noms et numéros de téléphone dans les tableaux (nous verrons plus tard comment les retrouver). Ajoutons les lignes :

```
40 FOR x=1 TO 100
50 INPUT "entrez le nom";NOM$(x)
60 INPUT "entrez le numero de telephone";TEL$(x)
70 NEXT
run
```

Tout cela est fort bien, mais nous n'avons pas l'intention d'entrer les 100 noms en une seule fois. De plus, la présentation du programme à l'écran laisse beaucoup à désirer. Il s'agit maintenant d'y mettre un peu d'ordre. Pour commencer, nous allons, avant chaque nouvelle entrée, débarrasser l'écran du texte antérieur devenu inutile. C'est l'affaire de l'instruction **CLS** :

```
45 CLS
```

Mais comment indiquer à l'ordinateur que, pour le moment, nous avons fini d'introduire des données ? Bien sûr, on peut toujours arrêter un programme en frappant la touche **[ESC]**, mais alors nos précieux enregistrements seront effacés dès que nous donnerons à nouveau l'ordre **RUN** !

Il existe une meilleure solution : à chaque enregistrement d'un nouveau nom, demandons au programme de vérifier si quelque chose a effectivement été frappé au clavier, c'est à dire si **NOM$(x)** n'est pas une chaîne vide, et demandons lui de s'interrompre dans le cas contraire. Avez-vous deviné par quel biais ? Ecrivons :

```
55 IF NOM$(x)="" THEN 80
80 PRINT"entree des donnees terminee"
```

Le programme doit lui-même indiquer à l'utilisateur comment l'interrompre. Ajoutons donc :

```
47 PRINT"pour arreter appuyer [ENTER]"
```

Voyons maintenant comment obtenir l'impression des informations enregistrées, d'abord sous forme de liste. Ecrivons :

```
90 FOR x=1 TO 100
100 PRINT NOM$(x);" ";TEL$(x)
110 NEXT
```

Mais là encore le programme ne sait pas comment s'arrêter avant le centième élément du tableau. Ajoutons donc :

```
95 IF NOM$(x)="" THEN 120
120 PRINT"liste terminee"
```

La chaîne vide est détectée en ligne **95** et le programme interrompt l'impression en sautant les lignes **100** et **110**.

Passons maintenant à notre objectif suivant : nous voulons que le programme soit capable de rechercher un nom entré au clavier. Ecrivons :

```
130 INPUT "nom a trouver :";RECHERCHE$
140 FOR x=1 TO 100
150 IF INSTR(NOM$(x),RECHERCHE$)=0 THEN 180
160 PRINT NOM$(x);" ";TEL$(x)
170 END
180 NEXT
190 PRINT "Ce nom n'est pas repertorie"
run
```

Une nouvelle instruction est apparue en ligne **15Ø : INSTR.** Cette instruction demande à l'ordinateur de chercher à l'intérieur de la première chaîne citée, la première occurrence de la seconde chaîne. Autrement dit, il va chercher dans **NOM$** une occurrence de **RECHERCHE$** (la variable introduite en ligne **13Ø**, contenant le nom que vous recherchez). Si **INSTR** ne trouve pas au moins une partie de cette chaîne, elle fournira la valeur 0, ce qui dans notre programme provoque un saut en ligne **18Ø** et donc une nouvelle tentative avec la valeur suivante de **x (NEXT)**. Si le programme a balayé toutes les valeurs autorisées de x (1 à 100), il peut alors atteindre la ligne **19Ø** et annoncer qu'il n'a pas trouvé le nom demandé. S'il le trouve, **INSTR** fournit une valeur différente de 0. Le programme passe alors sur la ligne **16Ø**, affichant le nom et le numéro de téléphone, avant de s'achever à la ligne **17Ø** (END).

Notre programme se perfectionne très rapidement, mais il nous reste encore beaucoup à faire. Prenons donc un peu de recul et considérons ses limites. La façon dont il se déroule, par exemple : on enregistre les informations, puis on obtient une liste, et l'on peut enfin demander la recherche d'un nom donné.

Et si jamais ?

Et si jamais cet ordre ne vous convenait pas ? Si vous préfériez commencer par la recherche d'un nom enregistré la veille, ou bien ajouter des noms et des numéros de téléphone à ceux qui existent déjà ? Etudier et résoudre ce type de problème, c'est cela la programmation. BASIC, vous le savez, a la bonté de vous permettre l'insertion de nouvelles instructions dans un programme existant, mais un bon programmeur ne doit pas se laisser surprendre.

Un autre inconvénient, de taille, tient au fait que le programme stocke le contenu des tableaux dans une partie de la mémoire effacée à chaque nouvelle exécution. Comme il est hors de question que vous entriez à chaque fois toutes vos informations à la main, il faudra vous réserver la possibilité, d'une part, de sauvegarder les valeurs des variables **NOM$** et **TEL$** avant d'éteindre votre ordinateur et, d'autre part, de charger ces valeurs lors de l'exécution du programme.

Réponses

Voici comment nous allons résoudre le problème du déroulement des tâches : nous allons faire en sorte qu'en début d'exécution, le programme nous propose un choix entre les différentes opérations qu'il est capable d'effectuer. Ce programme, de type « piloté par menu », affiche effectivement à l'écran un menu des options proposées. Vous êtes-vous déjà servi d'un distributeur automatique de billets ? Si oui, vous avez déjà eu affaire à un programme piloté par menu. Incorporons ce menu à notre programme :

```
32 PRINT"1. ajouter un correspondant"
33 PRINT"2. lister les correspondants"
34 PRINT"3. consulter le repertoire"
35 PRINT"4. sauvegarder le repertoire"
36 PRINT"5. charger le repertoire"
37 INPUT "votre choix (puis ENTER)";ch
38 ON ch GOSUB 40,90,130
```

```
85 RETURN
125 RETURN
170 RETURN
200 RETURN
```

Comme vous pouvez le constater, le programme affiche maintenant le menu des options, puis introduit dans la variable **ch** le numéro entré au clavier (**INPUT**). En passant sur l'instruction **ON ch GOSUB** (ligne **38**), il lancera le premier sous-programme (ligne **38**) si ch = 1, le deuxième (ligne **9Ø**) si ch = 2 et ainsi de suite.

Maintenant que chacune des fonctions est devenue un sous-programme, il faut obligatoirement en indiquer la fin par une instruction **RETURN**, ce que nous avons fait.

Vous souvenez-vous du mode d'exécution de **RETURN** ? A la fin du sous-programme, cette instruction BASIC renvoie le programme à la ligne suivant immédiatement l'instruction **GOSUB** correspondante, c'est-à-dire, dans notre cas, à la ligne située après **38** (notre programme se poursuivrait donc à partir du point d'entrée des informations, ligne **4Ø**). Pour éviter cela, introduisons la ligne :

```
39 GOTO 32
```

...afin de faire boucler le programme sur l'affichage du menu. Faites encore tourner le programme pour constater les améliorations apportées.

Parfait ! Voyons maintenant comment notre programme se présente (si celui-ci n'est pas arrêté, faites **[ESC]** deux fois. Tapez :

```
LIST
```

Et voici ce que vous devriez avoir sous les yeux :

```
10 REM repertoire telephonique
20 DIM NOM$(100)
30 DIM TEL$(100)
32 PRINT"1. ajouter un correspondant"
33 PRINT"2. lister les correspondants"
34 PRINT"3. consulter le repertoire"
35 PRINT"4. sauvegarder le repertoire"
```

```
36 PRINT"5. charger le repertoire"
37 INPUT"votre choix (puis ENTER)";ch
38 ON ch GOSUB 40,90,130
39 GOTO 32
40 FOR x=1 TO 100
45 CLS
47 PRINT"pour arreter appuyer [ENTER]"
50 INPUT "nom";NOM$(x)
55 IF NOM$(x)="" THEN 80
60 INPUT "telephone";TEL$(x)
70 NEXT
80 PRINT"entree des donnees terminee."
85 RETURN
90 FOR x=1 TO 100
95 IF NOM$(x)="" THEN 120
100 PRINT NOM$(x);" ";TEL$(x)
110 NEXT
120 PRINT"fin de liste"
125 RETURN
130 INPUT "nom a trouver";RECHERCHE$
140 FOR x=1 TO 100
150 IF INSTR(NOM$(x),RECHERCHE$)=0 THEN 180
160 PRINT NOM$(x);" ";TEL$(x)
170 RETURN
180 NEXT
190 PRINT"ce nom n'est pas repertorie."
200 RETURN
```

Vous remarquez qu'à certains endroits nous commençons à manquer de place pour insérer de nouvelles lignes. Nous allons en créer et remettre un peu d'ordre en **RENUM**-érotant les lignes. Faites :

```
RENUM
LIST
```

Vous devez maintenant obtenir :

```
10 REM repertoire telephonique
20 DIM NOM$(100)
30 DIM TEL$(100)
40 PRINT"1. ajouter un correspondant"
50 PRINT"2. lister les correspondants"
60 PRINT"3. consulter le repertoire"
70 PRINT"4. sauvegarder le repertoire"
80 PRINT"5. charger le repertoire"
90 INPUT"votre choix (puis ENTER)";ch
```

```
100 ON ch GOSUB 120,210,270
110 GOTO 40
120 FOR x=1 TO 100
130 CLS
140 PRINT"pour arreter appuyer [ENTER]"
150 INPUT "nom";NOM$(x)
160 IF NOM$(x)="" THEN 190
170 INPUT "telephone";TEL$(x)
180 NEXT
190 PRINT"entree des donnees terminee."
200 RETURN
210 FOR x=1 TO 100
220 IF NOM$(x)="" THEN 250
230 PRINT NOM$(x);" ";TEL$(x)
240 NEXT
250 PRINT"fin de liste"
260 RETURN
270 INPUT "nom a trouver";RECHERCHE$
280 FOR x=1 TO 100
290 IF INSTR(NOM$(x),RECHERCHE$)=0 THEN 320
300 PRINT NOM$(x);" ";TEL$(x)
310 RETURN
320 NEXT
330 PRINT"ce nom n'est pas repertorie."
340 RETURN
```

Voilà qui est mieux. Continuons ! Il nous faut maintenant une instruction qui fera en sorte que chaque nouvelle information enregistrée soit rangée à l'intérieur de la première case vide disponible dans le tableau. Nous allons pour cela nous servir d'une nouvelle intruction : **LEN**. Celle-ci permet de calculer la longueur d'une chaîne. Voici ce qu'il faut indiquer à l'ordinateur :

Si (**IF**) la longueur (**LEN**gth) de **NOM$(x)** est supérieure à 0, autrement dit si cette case est déjà occupée, il faut alors (**THEN**) passer directement à la ligne **180** (qui donnera la référence de la case suivante).

Nul besoin, on le voit, d'être doué en anglais pour parler BASIC. Toutefois, c'est avant tout une question de bon sens !

```
135 IF LEN(NOM$(x))>0 THEN 180
```

Enfantin, n'est-ce pas ? Avec votre liste des mots-clés de BASIC, et un peu de réflexion, aucune difficulté ne pourra vous arrêter. Vous trouverez toujours une instruction répondant exactement à vos besoins et, avec l'habitude, les solutions vous viendront à l'esprit comme par miracle.

Voyons maintenant comment sauvegarder les valeurs des variables afin de pouvoir les recharger lorsqu'on exécute le programme. Vous savez, depuis la partie 7 du Cours Elémentaire, que l'on peut sauvegarder le programme lui-même au moyen de l'instruction **SAVE**. Cependant, ce programme n'est qu'un cadre permettant l'entrée des valeurs (au clavier) et leur sortie (sur l'écran). En sauvegardant le programme (**SAVE**), vous conservez ce cadre, mais pas les valeurs elles-mêmes.

Une partie du programme devra donc prévoir le stockage des valeurs sur disquette. Nous allons pour cela créer un « fichier de données » indépendant.

Il faudra d'abord ouvrir (**OPEN**) un fichier sortie (**OUT**put) que nous appellerons, par exemple, « **données** », puis écrire (**WRITE**) à l'intérieur de ce fichier les valeurs (de 1 à 100) des variables **NOM$(x)** et **TEL$(x)** et, pour finir, refermer (**CLOSE**) ce fichier avant de revenir au menu. Cette section sera placée à partir de la ligne **350**. Laissons à l'instruction :

```
AUTO 350
```

...le soin de numéroter **AUTO**matiquement les lignes :

```
350 OPENOUT "donnees"
360 FOR x=1 TO 100
370 WRITE #9,NOM$(x),TEL$(x)
380 NEXT
390 CLOSEOUT
400 PRINT"donnees sauvees"
410 RETURN
```

Une fois la ligne **410** entrée par **[RETURN]**, faites **[ESC]** pour interrompre la numérotation **AUTO**matique.

Nous venons d'introduire une option supplémentaire : il nous faut donc ajouter un numéro dans la liste située après l'instruction **ON ch GOSUB**, ligne **100**. Rappelons la ligne **100** pour opérer cette modification, par **EDIT** :

```
100 ON ch GOSUB 120,210,270,350
```

Vous pourrez désormais, en choisissant l'option 4, demander au programme de stocker sur disquette les informations introduites.

Vous remarquerez qu'au moment où le programme écrit sur la disquette les valeurs de **NOM$(x)** et de **TEL$(x)**, le mot clé **WRITE** est suivi de l'expression # 9. Le symbole # est un « sélecteur de canal » qui indique à l'ordinateur sur quel canal les données doivent être acheminées. Il en existe 10 dans un ordinateur :

Si l'on dirige les données sur les canaux 0 à 7 (de #**0** à #**7**), celles-ci apparaissent sur l'écran (les canaux **0** à **7** sont réservés pour l'écran et y définissent des « fenêtres »).

Les données envoyées sur le canal # 8 sortiront sur l'imprimante, le cas échant.

Enfin, les données dirigées sur le canal # 9 seront envoyées à l'unité de disques : c'est ce que nous avons fait ligne **370**.

Une petite digression...

Juste un mot sur l'instruction AUTO que nous venons d'utiliser. Lorsque l'on entre, sans numéro de ligne, l'instruction :

```
AUTO
```

...l'ordinateur se met à numéroter automatiquement les lignes, en commençant à 10, et en progressant de 10 en 10 à chaque nouvelle ligne entrée. Si les lignes **10, 20, 30**, etc. existent déjà dans votre programme, vous les ferez apparaître sur l'écran une à une en actionnant la touche **[RETURN]**. Cette procédure peut vous faire gagner du temps lorsque vous désirez corriger toute une séquence de lignes espacées de 10 en 10.

Revenons à notre programme...

Après avoir écrit les instructions permettant le stockage de nos données sur disquette, il ne nous reste plus qu'à en prévoir le chargement, pour que notre programme soit terminé. Ajoutons donc une nouvelle option au menu en corrigeant encore une fois la ligne **100** :

```
100 ON ch GOSUB 120,210,270,350,420
```

Voyons maintenant les instructions de chargement. Il faudra d'abord ouvrir (**OPEN**) le fichier entrée (**IN**put) sur la disquette de « **données** », puis chercher à partir de cette disquette (canal # 9) toutes les valeurs, de 1 à 100, des variables **NOM$(x)** et **TEL$(x)**, et enfin refermer le fichier avant de revenir au menu. Voici la procédure :

```
420 OPENIN "donnees"
430 FOR x=1 TO 100
440 INPUT #9,NOM$(x),TEL$(x)
450 NEXT
460 CLOSEIN
470 PRINT"donnees chargees"
480 RETURN
```

La fin du commencement...

Le programme que nous venons d'écrire remplit maintenant tous les objectifs que nous nous étions fixés au départ lorsque que nous nous demandions « ce qu'il devrait faire ». Il ne nous reste donc plus qu'à améliorer la « présentation des résultats » à l'écran.

Et le commencement de la fin...

D'abord quelques instructions pour mettre le programme au propre :

```
34 MODE 1
```

Ceci détermine le mode d'affichage : l'écran sera effacé à chaque nouvelle exécution du programme. Ecrivez ensuite :

```
36 WINDOW #1,7,36,10,14
```

Cette instruction un peu obscure ne doit pas vous effrayer : elle consiste simplement à dessiner sur l'écran une petite fenêtre pour encadrer le menu. Le code situé après le mot **WINDOW** indique à l'ordinateur sur quel canal cette fenêtre sera dirigée (souvenez-vous que nous en avons 8 à notre disposition, de #**0** à #**7**). Sachant de plus que l'ordinateur, s'il n'a pas reçu d'indication particulière, choisit automatiquement le canal #**0**, on voit bien qu'il faut éviter de diriger notre petite fenêtre sur ce canal, sous peine de voir s'y afficher toutes les sorties du programme. Il faudra donc choisir un autre canal entre #**1** et #**7**, d'où l'indication #**1** sur notre exemple. Les quatre nombres qui viennent ensuite indiquent, de manière on ne peut plus simple, les dimensions de la fenêtre : ils donnent la position des bords gauche, droit, supérieur et inférieur de la fenêtre, en se référant aux numéros de colonnes et de lignes de l'écran (comme pour l'instruction **LOCATE**). Ainsi dans notre exemple, après avoir précisé que nous utilisions le canal 1, nous déclarons que le bord de gauche commence en colonne 7, que celui de droite finit en colonne **36**, que le bord supérieur commence en ligne **10** et que le bord inférieur finit en ligne **14**.

Si nous voulons maintenant que le menu s'affiche dans cette fenêtre, il va nous falloir rectifier les lignes **40** à **80** :

```
40 PRINT #1,"1. ajouter un correspondant"
50 PRINT #1,"2. lister les correspondants"
60 PRINT #1,"3. consulter le repertoire"
70 PRINT #1,"4. sauvegarder le repertoire"
80 PRINT #1,"5. charger le repertoire"
```

Ajoutons encore la ligne :

```
85 LOCATE 7,25
```

Cette instruction sert à positionner sur l'écran la demande de l'option choisie. La présentation sera ainsi plus claire.

Vidons maintenant l'écran de son contenu avant chaque retour au menu :

```
110 GOTO 34
```

...et après chaque selection d'option :

```
95 CLS
```

Ajoutons pour finir les trois lignes suivantes, qui mettront l'ordinateur en attente avant de revenir au menu :

```
103 LOCATE 7,25
105 PRINT"appuyez une touche pour le menu"
107 IF INKEY$="" THEN 107
```

La ligne **103** indique l'endroit auquel l'ordinateur devra afficher le message contenu en ligne **105**. La ligne **107** cherche quelle chaîne de caractère vient d'être entrée au clavier. Tant que celle-ci sera vide, c'est-à-dire tant que l'utilisateur n'aura pas actionné une touche quelconque du clavier, le programme bouclera sur cette instruction. Cette instruction a bien pour effet de mettre le programme en attente : celui-ci attend effectivement qu'une touche soit enfoncée pour passer à la ligne suivante.

Le voilà donc terminé, ce programme ! Terminé, vraiment ? ...vous pourriez encore lui demander de corriger ou d'effacer des noms et des numéros de téléphone, de trier la liste dans l'ordre alphabétique, ou de vous la « sortir » sur imprimante, ou bien encore, si vous êtes très ambitieux, d'émettre des signaux permettant d'appeler automatiquement votre correspondant en entrant simplement son nom au clavier, non sans avoir, bien entendu, demandé aux PTT l'autorisation de connecter votre ordinateur au poste téléphonique ! Ces perfectionnements sont pourtant de l'ordre du possible. A vrai dire, on peut améliorer et peaufiner ainsi ses programme à l'infini, surtout lorsqu'on dispose d'un outil informatique aussi puissant que le 6128. Il faut pourtant savoir s'arrêter et nous allons laisser ce « répertoire téléphonique » à ce point en espérant que vous en savez désormais un peu plus sur l'art d'écrire un programme en partant de zéro. Il ne vous reste qu'à le remettre un peu en ordre en tapant une dernière instruction :

```
RENUM
```

...puis à le stocker sur disquette, ou vous en débarrasser. Mais qui sait, il pourrait peut-être vous être utile... pour noter les numéros de téléphone de vos amis !

Voici votre programme terminé :

```
10 REM repertoire telephonique
20 DIM NOM$(100)
30 DIM TEL$(100)
40 MODE 1
50 WINDOW #1,7,36,10,14
60 PRINT #1,"1. ajouter un correspondant"
70 PRINT #1,"2. lister les correspondants"
80 PRINT #1,"3. consulter le repertoire"
90 PRINT #1,"4. sauvegarder le repertoire"
100 PRINT #1,"5. charger le repertoire"
110 LOCATE 7,25
120 INPUT"votre choix (puis ENTER)";ch
130 CLS
140 ON ch GOSUB 190,290,350,430,500
150 LOCATE 5,25
160 PRINT"appuyer une touche pour le menu"
170 IF INKEY$="" THEN 170
180 GOTO 40
190 FOR x=1 TO 100
200 CLS
210 IF LEN(NOM$(x))>0 THEN 260
220 PRINT"pour arreter appuyer [ENTER]"
230 INPUT "nom";NOM$(x)
240 IF NOM$(x)="" THEN 270
250 INPUT "telephone";TEL$(x)
260 NEXT
270 PRINT"entree des donnees terminee."
280 RETURN
290 FOR x=1 TO 100
300 IF NOM$(x)="" THEN 330
310 PRINT NOM$(x);" ";TEL$(x)
320 NEXT
330 PRINT"fin de liste"
340 RETURN
350 INPUT "nom a trouver";RECHERCHE$
360 FOR x=1 TO 100
370 IF INSTR(NOM$(x),RECHERCHE$)=0 THEN 400
380 PRINT NOM$(x);" ";TEL$(x)
390 RETURN
400 NEXT
410 PRINT"ce nom n'est pas repertorie."
420 RETURN
430 OPENOUT "donnees"
```

```
440 FOR x=1 TO 100
450 WRITE #9,NOM$(x),TEL$(x)
460 NEXT
470 CLOSEOUT
480 PRINT"donnees sauvees"
490 RETURN
500 OPENIN "donnees"
510 FOR x=1 TO 100
520 INPUT #9,NOM$(x),TEL$(x)
530 NEXT
540 CLOSEIN
550 PRINT"donnees chargees"
560 RETURN
run
```

Chapitre 3
Liste complète des mots clés du BASIC Schneider CPC6128

IMPORTANT

L'assimilation de la terminologie et de la notation adoptées dans ce chapitre est indispensable. Différents types de parenthèses expliquent l'entrée des commandes ; commencez donc par vous familiariser avec la signification de chacun.

Les commandes sans crochets doivent rester telles quelles. C'est le cas, par exemple, de la commande :

```
END
```

...et vous devez taper le mot **END** en toutes lettres.

Lorsqu'un élément apparaît entre parenthèses angulaires, < > comme :

<numéro de ligne>

...il ne faut ni taper ces parenthèses, ni ce qu'elles renferment. L'exemple ci-dessus indique le type de données requises pour la commande. Par exemple :

EDIT <numéro de ligne>

...signifie que vous devez taper :

```
EDIT 100
```

Les parenthèses **()** DOIVENT être tapées telles quelles. Dans l'exemple :

COS (<expression numérique>)

...l'expression numérique doit se trouver entre parenthèses. Par exemple :

```
PRINT COS(45)
```

Les crochets, pour finir, contiennent les éléments facultatifs d'une commande ou d'une fonction. Par exemple :

RUN [<numéro de ligne>]

...signifie que vous n'êtes pas obligé de faire suivre le mot clé **RUN** d'un paramètre, mais que vous pouvez, si vous le désirez, ajouter le paramètre <numéro de ligne>. La commande peut alors être entrée de deux manières :

```
RUN ...ou... RUN 100
```

Caractères spéciaux

& ou **&H** Préfixe pour un nombre hexadécimal
&X Préfixe pour un nombre binaire
Préfixe pour un canal d'entrées-sorties

Types de données

Les chaînes peuvent avoir de 0 à 255 caractères et sont désignées par <chaîne alphanumérique>. On peut adjoindre une chaîne à une autre avec le signe + et à condition que la chaîne qui en résulte ne dépasse pas 255 caractères.

Les entiers varient de -32768 à +32767 et les nombres réels de -1.7E+38 à +1.7E+38, le plus petit nombre au dessus de zéro étant 2.9E-39 et chaque nombre ayant 9 à 10 chiffres significatifs.

Une <expression numérique> est une expression qui aboutit à une valeur numérique. Cela peut être des nombres, ou une variable numérique, ou des nombres opérant avec des variables numériques, à peu près tout ce qui n'est pas <chaîne alphanumérique>.

Un <numéro de canal> se rapporte à une <expression numérique> pour désigner une fenêtre d'écran, une imprimante, une cassette ou une disquette vers laquelle le texte doit être acheminé.

Une <liste de:élément> décrit un paramètre comprenant une liste d'éléments séparés par des virgules. Cette liste, pouvant contenir un ou plusieurs éléments, est limitée par la longueur de ligne.

Les différents indicateurs de type de données :

% Entiers
! Réels (par défaut)
$ Chaîne alphabétique et numérique (alphanumérique)

Nous allons donner les mots clés du BASIC Schneider sous la forme suivante :

MOT CLE

Syntaxe

Exemple

Description

Mots clés associés

Les mots clés sont :

DES COMMANDES : opérations exécutées directement
DES FONCTIONS : opérations intervenant dans une expression
DES OPERATEURS : agissent sur des arguments mathématiques.

Lors du listage d'un programme, le BASIC transforme en MAJUSCULES tous les mots clés tapés en minuscules. Les exemples de ce chapitre sont en MAJUSCULES tels qu'ils apparaissent après un **LIST**ing. Il sera toutefois préférable que vous les tapiez en minuscules car les mots clés contenant une erreur resteront **LIST**és en minuscules, vous révélant ainsi vos erreurs de frappe.

Pour plus de détails sur le BASIC AMSTRAD 6128, consultez les spécifications SOFT967.

Mots clés...

ABS

ABS (<expression numérique>)

```
PRINT ABS(-67.98)
 67.98
```

FONCTION : Donne la valeur **ABS**olue de l'expression entre parenthèses. Les nombres négatifs perdent donc leur signe moins.

Mots clés associés : **SGN**

AFTER

AFTER <délai du chronomètre>[,<numéro de chronomètre>] **GOSUB** <numéro de ligne>

```
10 AFTER 250 GOSUB 60:CLS
20 PRINT"Devine une lettre en 5 secondes"
30 a$=INKEY$:IF flag=1 THEN END
40 IF a$<>CHR$(INT(RND*26+97)) THEN 30
50 PRINT a$;" est exacte.tu as gagne !"
55 SOUND 1,478:SOUND 1,358:END
60 PRINT"Trop tard.J'ai gagne !"
70 SOUND 1,2000:flag=1:RETURN
run
```

COMMANDE : Appelle un sous-programme après (= **AFTER** en anglais) un certain délai. Le <délai du chronomètre> indique la durée de l'attente en multiples de 0,02 seconde. Le <numéro de chronomètre> (qui peut être 0, 1, 2 ou 3) précise lequel des quatre chronomètres d'attente il faut utiliser.

Chacun des 4 chronomètres peut être associé à un sous-programme. Pour plus de détails concernant les interruptions, reportez-vous à la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **EVERY**, **REMAIN**, **RETURN**

AND

<argument> AND <argument>

```
IF "alain" < "bernard" AND "chien" > "chat"  THEN  PRINT
 "vrai" ELSE PRINT "faux"
vrai
IF "bernard" < "alain" AND "chat" > "chien"  THEN  PRINT
 "vrai" ELSE PRINT "faux"
faux
IF "alain" < "bernard" AND "chat" > "chien"  THEN  PRINT
 "vrai" ELSE PRINT "faux"
faux
....
PRINT 1 AND 1
1
PRINT 0 AND 0
0
PRINT 1 AND 0
0
```

OPERATEUR : Exécute des opérations booléennes par bits sur des nombres entiers. Est égal à **0** sauf lorsque les deux bits d'arguments sont égaux à 1.

Pour toute information complémentaire sur la logique, consultez la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **OR**, **NOT**, **XOR**.

ASC

ASC (<chaîne alphanumérique>)

```
PRINT ASC("x")
 120
```

FONCTION : Donne la valeur numérique du premier caractère d'une chaîne de caractères.

Mots clés associés : **CHR$**

ATN

ATN (<expression numérique>)

```
PRINT ATN(1)
 0.785398163
```

FONCTION : Calcule l'**A**rc **T**a**N**gente (réduisant l'<expression numérique> à un nombre réel en radians compris entre -PI/2 et +PI/2) de la valeur donnée.

A noter : les commandes **DEG** et **RAD** peuvent être utilisées pour spécifier explicitement que le résultat sera exprimé respectivement en degrés ou radians.

Mots clés associés : **COS**, **DEG**, **RAD**, **SIN**, **TAN**

AUTO

AUTO [<numéro de ligne>][,<incrément>]

```
AUTO 100,50
```

COMMANDE : Génère **AUTO**matiquement les numéros de ligne. Le paramètre facultatif <numéro de ligne> donne le premier numéro de ligne à générer. Si vous ne le précisez pas, les lignes sont générées à partir de **10**.

L'<incrément>, également facultatif, fixe l'intervalle entre les numéros de ligne. En l'absence de spécification, il sera égal à **10**. Si un numéro de ligne déjà utilisé est généré, le contenu de cette ligne apparaît à l'écran et peut éventuellement être modifié. La ligne affichée est ensuite remplacée en mémoire après activation de la touche **[RETURN]**. Pour arrêter la numérotation automatique des lignes, appuyez sur **[ESC]**.

Mots clés associés : Aucun

BIN$

BIN$ (<nombre entier sans signe>[,<nombre entier>])

```
PRINT BIN$(64,8)
01000000
```

FONCTION : Produit une chaîne de chiffres **BIN**aires représentant la valeur du <nombre entier sans signe>, à l'aide du nombre de chiffres binaires indiqué par le deuxième <nombre entier> (entre 0 et 16). Si ce nombre est trop grand, le résultat commence par autant de zéros que nécessaire. S'il est trop petit, le résultat n'est pas tronqué mais converti en autant de chiffres que nécessaire.

Le <nombre entier sans signe> à convertir en binaire doit être compris entre -32768 et 65535.

Mots clés associés : **DEC$**, **HEX$**, **STR$**

BORDER

BORDER <numéro de couleur>[,<numéro de couleur>]

```
10 REM 729 combinaisons de bordure
20 SPEED INK 5,5
30 FOR a=0 TO 26
40 FOR b=0 TO 26
50 BORDER a,b:CLS:LOCATE 14,13
60 PRINT"border";a;",";b
70 FOR t=1 TO 500
80 NEXT t,b,a
run
```

COMMANDE : Pour changer la couleur de la bordure d'écran. Si deux couleurs sont indiquées, elles alternent à la vitesse déterminée par la commande **SPEED INK**, le cas échéant. Les valeurs vont de 0 à 26.

Mots clés associés : **SPEED INK**

BREAK

(Voir **ON BREAK CONT**, **ON BREAK GOSUB**, **ON BREAK STOP**)

CALL

CALL <adresse>[,<liste de:<paramètre>]

```
CALL 0
```

COMMANDE : Permet à un sous-programme externe d'être appelé à partir du BASIC. L'exemple ci-dessus réinitialise complètement l'ordinateur.

A utiliser avec précaution.

Mots clés associés : **UNT**

CAT

CAT

```
CAT
```

COMMANDE : Demande au BASIC de lire le **CAT**alogue de la disquette. Affiche en ordre alphahumérique les noms de tous le fichiers présents ainsi que leur longueur (en arrondissant au Koctet supérieur). Le nombre d'octets disponibles s'affiche également avec l'indentification de la disquette et de l'utilisateur.

Cette commande n'a pas d'effet sur le programme en cours.

Mots clés associés : **LOAD**, **RUN**, **SAVE**

CHAIN

CHAIN <nomfich>[,<numéro de ligne>]

```
CHAIN "testprog.bas",350
```

COMMANDE : Charge un programme en mémoire à partir d'une disquette, remplaçant le programme existant. Le nouveau programme commence au début ou à partir de la ligne spécifiée par <numéro de ligne>.

Les fichiers protégés (sauvegardés par la commande **SAVE,p**) peuvent être chargés et lancés par **CHAIN**.

Mots clés associés : **CHAIN MERGE**, **LOAD**, **MERGE**

CHAIN MERGE

CHAIN MERGE <nomfich>[,<numéro de ligne>]
[,**DELETE** <ensemble de lignes>]

```
CHAIN MERGE "partie2.bas",750,DELETE 400-680
```

COMMANDE : Charge en mémoire un programme sur disquette en le fusionnant au programme existant, puis lance le programme résultant, depuis le début ou à partir de la ligne spécifiée par le paramètre en option, <numéro de ligne>. Si vous désirez effacer une partie du programme initial avant d'utiliser cette commande, vous pouvez spécifier le paramètre <ensemble de lignes> de la commande **DELETE**. Les numéros de ligne du programme initial identiques à ceux du programme à fusionner seront écrasées sur les nouvelles lignes. Les fichiers protégés (sauvegardés par la commande **SAVE,p**) ne peuvent PAS être fusionnés puis lancés par cette commande.

Mots clés associés : **LOAD**, **MERGE**, **DELETE**, **CHAIN**

CHR$

CHR$ (<nombre entier>)

```
10 FOR x=32 TO 255
20 PRINT x;CHR$(x),
30 NEXT
run
```

FONCTION : Convertit un <nombre entier> compris entre 0 et 255 en une chaîne de caractères équivalente à l'aide du jeu de caractères de l'AMSTRAD 6128, décrit dans la partie 3 du chapitre « Pour information... ». Les caractères 0 à 31 sont des caractères de contrôle. C'est pourquoi l'exemple ci-dessus affiche les entiers compris entre 32 et 255.

Mots clés associés : **ASC**

CINT

CINT (<expression numérique>)

```
10 n=1.9999
20 PRINT CINT(n)
run
 2
```

FONCTION : Convertit une valeur numérique en un entier arrondi compris entre -32768 et 32767.

Mots clés associés : **CREAL**, **FIX**, **INT**, **ROUND**, **UNT**

CLEAR

CLEAR

```
CLEAR
```

COMMANDE : Efface toutes les variables, fichiers ouverts, tableaux et fonctions utilisateur, le mode de calcul en BASIC s'effectue en radians.

Mots clés associés : Aucun

CLEAR INPUT

CLEAR INPUT

```
10 CLS
20 PRINT" tapez plusieurs lettres maintenant !"
30 FOR t=1 TO 3000
40 NEXT
50 CLEAR INPUT
run
```

COMMANDE : Efface toutes les données entrées à partir du clavier, se trouvant dans le tampon.

Pour expérimenter cette commande, lancez le programme ci-dessus et tapez les lettres lorsque vous y êtes invité. Supprimez ensuite la ligne **50** du programme, relancez-le et voyez la différence.

Mots clés associés : **INKEY**, **INKEY$**, **JOY**

CLG

CLG [<encre>]

```
LOCATE 1,20
CLG 3
```

COMMANDE : Efface l'écran graphique et le ramène à sa couleur de fond. Si <encre> est spécifiée, le fond est de la couleur fixée en accord.

Mots clés associés : **CLS**, **GRAPHICS PAPER**, **INK**, **ORIGIN**

CLOSEIN

CLOSEIN

```
CLOSEIN
```

COMMANDE : Ferme tout fichier d'entrée ouvert sur la disquette (voir **OPENIN**).

Mots clés associés : **EOF**, **OPENIN**

CLOSEOUT

CLOSEOUT

```
CLOSEOUT
```

COMMANDE : Ferme tout fichier ouvert en sortie sur la disquette. (voir **OPENOUT**).

Mots clés associés : **OPENOUT**

CLS

CLS [# <numéro de canal>]

```
10 PAPER #2,3
20 CLS #2
run
```

COMMANDE : Efface la fenêtre d'écran spécifiée par le <numéro de canal> et lui donne sa couleur de papier. En l'absence de <numéro de canal>, **Ø** est pris par défaut.

Mots clés associés : **CLG**, **INK**, **PAPER**, **WINDOW**

CONT

CONT

```
CONT
```

COMMANDE : **CONT**inue l'exécution du programme après un **STOP**, ou deux activations de la touche **[ESC]**, si le programme n'a été ni modifié ni protégé. Des commandes directes peuvent être tapées avant reprise du programme.

Mots clés associés : **STOP**

COPYCHR$

COPYCHR$ (# <numéro de canal>)

```
10 CLS
20 PRINT"coin superieur"
30 LOCATE 1,1
40 a$=COPYCHR$(#0)
50 LOCATE 1,20
60 PRINT a$
run
```

FONCTION : Copie un caractère à partir de la position du curseur dans le canal (qui DOIT être spécifié). Le programme ci-dessus copie un caractère de l'emplacement 1,1 (angle supérieur gauche) et le reproduit en 1,20. Si le caractère lu n'est pas reconnu, une chaîne nulle est renvoyée.

Mots clés associés : **LOCATE**

COS

COS (<expression numérique>)

```
DEG
PRINT COS(45)
 0.707106781
```

FONCTION : Calcule le **COS**inus de l'<expression numérique>.

DEG et **RAD** peuvent servir à exprimer l'argument en degrés ou en radians, respectivement.

Mots clés associés : **ATN**, **DEG**, **RAD**, **SIN**

CREAL

CREAL (<expression numérique>)

```
10 a=PI
20 PRINT CINT(a)
30 PRINT CREAL(a)
run
 3
 3.14159265
```

FONCTION : Convertit l'<expression numérique> en nombre réel.

Mots clés associés : **CINT**

CURSOR

CURSOR [<indicateur système>][,<indicateur utilisateur>]

```
10 CURSOR 1
20 PRINT"question ?";
30 a$=INKEY$:IF a$="" THEN 30
40 PRINT a$
50 CURSOR 0
run
```

COMMANDE : Active ou désactive l'indicateur système ou utilisateur. Les paramètres <indicateur système> et <indicateur utilisateur> doivent être sur **0** (inactif) ou 1 (actif). Dans la commande **INKEY$** ci-dessus, le curseur a été rendu visible par fixation de l'indicateur système sur 1 (à la ligne **10**).

Le curseur s'affiche lorsque les deux indicateurs sont sur 1. Le curseur système est automatiquement activé pour la commande **INPUT** et désactivé pour **INKEY$**.

Il est préférable de désactiver le curseur pour affichage d'un texte à l'écran.

Vous pouvez omettre l'un des indicateurs mais pas les deux. Si un paramètre est omis, son état est inchangé.

Mots clés associés : **LOCATE**

DATA

DATA <liste de:<constante>

```
10 FOR x=1 TO 4
20 READ nom$,prenom$
30 PRINT"Mr. ";nom$;" ";prenom$
40 NEXT
50 DATA DUPONT,Olivier,DURAND,Francois
60 DATA LAMIE,Frederic,MOULIN,Daniel
run
```

COMMANDE : Déclare des données constantes à l'intérieur d'un programme. Ces données peuvent être affectées à une variable par la commande **READ**, après quoi le pointeur passe à l'élément suivant de la liste **DATA**. La commande **RESTORE** peut servir à déplacer le pointeur sur une position spécifiée de **DATA**.

Pour de plus amples informations, consultez la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **READ**, **RESTORE**

DEC$

DEC$ (<expression numérique>,<modèle de format>)

```
PRINT DEC$(10↑7,"££#########,.##")
 £10,000,000.00
```

FONCTION : Donne une représentation **DEC**imale de l'expression numérique, utilisant le <modèle de format> indiqué.

Le modèle de format ne peut contenir QUE les caractères :

+ - £ $ * # , . ↑

L'emploi de ces « indicateurs de format » est décrit au mot clé **PRINT USING**.

Mots clés associés : **BIN$**, **HEX$**, **PRINT USING**, **STR$**

DEF FN

DEF FN <nom>[(<paramètres formels>)]=<expression>

```
10 t=TIME/300
20 DEF FNchrono=INT(TIME/300-t)
30 EVERY 100 GOSUB 50
40 GOTO 40
50 PRINT"Le programme tourne depuis";
60 PRINT FNchrono;"secondes"
70 RETURN
run
```

COMMANDE : Le BASIC permet au programme de **DEF**inir une **F**o**N**ction retournant une valeur unique et de l'utiliser. **DEF FN** est la partie définition du mécanisme de création d'une fonction spécifique, travaillant d'une manière similaire aux fonctions existantes du BASIC (**COS**, **SIN**, **ATN**, etc.).

(Dans l'exemple ci-dessus la valeur de la fonction **FNchrono** est constamment mise à jour, même si le programme est suspendu par **[ESC]** ou arrêté par double **[ESC]**, puis relancé.

Mots clés associés : Aucun

DEFINT

DEFINT <liste de:<lettres concernées>

```
10 DEFINT n
20 nombre=123.456
30 PRINT nombre
run
 123
```

COMMANDE : Définit le type de variable par **DEF**aut, le type étant entier. Lorsqu'une variable intervient sans marqueur (**! % $**), le type par défaut est automatiquement mis en œuvre. Cette commande définit le type par défaut des variables selon la première lettre du nom de la variable. Elle peut être suivie d'une liste d'initiales. Par exemple :

```
DEFINT a,b,c
```

...ou d'une fourchette d'initiales :

```
DEFINT a-z
```

Mots clés associés : **DEFREAL, DEFSTR**

DEFREAL

DEFREAL <liste de:<lettres concernées>

```
DEFREAL x,a-f
```

COMMANDE : Définit le type de variable par **DEF**aut, le type étant réel. Lorsqu'une variable intervient sans identificateur de type (**! % $**), le type par défaut est automatiquement mis en œuvre. Le type de la variable sera déterminé suivant la première lettre du nom de la variable. Elle peut être suivie d'une liste d'initiales :

```
DEFREAL a,b,c
```

...ou d'une fourchette d'initiales :

```
DEFREAL a-z
```

Mots clés associés : **DEFINIT, DEFSTR**

DEFSTR

DEFSTR <liste de:<lettres concernées>

```
10 DEFSTR n
20 nom="Amstrad"
30 PRINT nom
run
Amstrad
```

COMMANDE : Définit le type de variable par **DEF**aut, le type étant une chaîne. Lorsqu'une variable intervient sans identificateur (**! % $**) le type par défaut est automatiquement mis en œuvre. Le type de la variable est déterminé selon la première lettre de son nom. La commande peut être suivie d'une liste d'initiales :

```
DEFSTR a,b,c
```

...ou d'une fourchette d'initiales :

```
DEFSTR a-z
```

Mots clés associés : **DEFINT**, **DEFREAL**

DEG

DEG

```
DEG
```

COMMANDE : Etablit le mode de calcul en **DEG**rés. Par défaut, les fonctions **SIN**, **COS**, **TAN** et **ATN** considèrent que l'argument qui leur est transmis est exprimé en radians. La commande reste valable jusqu'à ce qu'on utilise les commandes **RAD** ou **NEW**, **CLEAR**, **LOAD**, **RUN**, etc.

Mots clés associés : **ATN**, **COS**, **RAD**, **SIN**, **TAN**

DELETE

DELETE <ensemble de lignes>

```
DELETE 100-200
```

COMMANDE : Efface une partie du programme défini dans l'<ensemble de lignes>.

Il n'est pas nécessaire d'indiquer la ligne de départ ou la ligne d'arrivée pour effacer le programme « depuis le début » ou « jusqu'à la fin ».

```
DELETE -200
```

...ou :

```
DELETE 50-
```

...ou :

```
DELETE
```

...qui efface la totalité du programme.

Mots clés associés : **CHAIN MERGE, RENUM**

DERR

DERR

```
LOAD "xyz.abc"

XYZ .ABC not found
Ready
PRINT DERR
 146
```

FONCTION : Rapporte le dernier code d'**ERR**eur envoyé par le système de gestion de la Disquette. La valeur de **DERR** peut servir à confirmer l'erreur détectée. Consultez la liste des messages d'erreur du chapitre « Pour information... ».

Mots clés associés : **ERL, ERR, ERROR, ON ERROR GOTO, RESUME**

DI

DI

```
10 CLS:TAG:EVERY 10 GOSUB 90
20 x1=RND*320:x2=RND*320
30 y=200+RND*200:c$=CHR$(RND*255)
40 FOR x=320-x1 TO 320*x2 STEP 4
50 DI
60 MOVE 320,0,1:MOVE x-2,y:MOVE x,y
70 PRINT" ";c$;:FRAME
80 EI:NEXT:GOTO 20
90 MOVE 320,0:DRAW x+8,y-16,1:RETURN
run
```

COMMANDE : Désactive une Interruption (autre que **[ESC])** jusqu'à ce qu'elle soit réactivée directement par une commande **EI** ou indirectement par un **[RETURN]** à la fin d'un sous-programme d'interruption **GOSUB**.

L'entrée dans un sous-programme d'interruption désactive automatiquement les interruptions de priorité égale ou inférieure.

On l'utilise quand le programme doit s'exécuter sans interruption, par exemple quand deux sous-programmes sont en compétition pour utiliser les ressources de l'ordinateur (les ressources graphiques dans le programme ci-dessus, par exemple).

Pour de plus amples informations sur les interruptions consultez la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **AFTER**, **EI**, **EVERY**, **REMAIN**

DIM

DIM <liste de:<variable indicée>

```
10 CLS
20 DIM ami$(5),tel$(5)
30 FOR n=1 TO 5
40 PRINT"telephone No";n
50 INPUT "entrez le nom";ami$(n)
60 INPUT "entrez le numero de tel.";tel$(n)
70 PRINT
80 NEXT
90 FOR n=1 TO 5
100 PRINT n;ami$(n),tel$(n)
110 NEXT
run
```

COMMANDE : **DIM**ensionne un tableau. Cette commande alloue l'espace requis aux tableaux et spécifie les valeurs d'indices maximales. Le BASIC doit connaître l'espace réservé pour un tableau, en l'absence de spécification il prend 10 comme valeur par défaut.

Un tableau est identifié par une <variable indicée>, à savoir un nom de variable accompagné d'un ensemble d'indices afin que chaque « élément » du tableau ait sa propre valeur d'indice.

Une boucle **FOR NEXT** peut servir à contrôler le tableau en traitant chaque élément du tableau à tour de rôle.

La valeur minimale d'un indice est zéro (c'est le premier élément d'un tableau).

Les tableaux peuvent être multi-dimensionnels et chaque élément est référencé par sa position. Par exemple, dans un tableau dimensionné par :

```
DIM position$(20,20,20)
```

...un élément du tableau sera référencé de la façon suivante :

```
position$(4,5,6)
```

Mots clés associés : **ERASE**

DRAW

DRAW <coordonnée x>,<coordonnée y> [,[<encre>][,<mode d'encre>]]

```
10 MODE 0:BORDER 0:PAPER 0:INK 0,0
20 x=RND*640:y=RND*400:z=RND*15
30 DRAW x,y,z
40 GOTO 20
run
```

COMMANDE : Trace une ligne sur l'écran entre la position du curseur graphique et une position absolue spécifiée par les coordonnées x et y. L'<encre> de traçage peut être spécifiée (entre 0 et 15).

Le <mode d'encre> facultatif détermine l'interaction de l'encre sur l'affichage présent à l'écran. Les quatre <modes d'encre> sont les suivants :

0: Normal
1: XOR (OU exclusif)
2: AND (ET)
3: OR (OU)

Mots clés associés : **DRAWR**, **GRAPHICS PEN**, **MASK**

DRAWR

DRAWR <décalage x>,<décalage y>[,[<encre>][,<mode d'encre>]]

```
10 CLS:PRINT"tu montes au premier !?"
20 MOVE 0,350:FOR n=1 TO 8
30 DRAWR 50,0
40 DRAWR 0,-50
50 NEXT:MOVE 348,0:FILL 3
60 GOTO 60
run
```

COMMANDE : Trace une ligne sur l'écran graphique à partir du curseur graphique jusqu'à la position spécifiée par les <décalages x et y>. L'<encre> du tracé peut être spécifiée (entre 0 et 15).

Le <mode d'encre> facultatif détermine l'interaction de l'encre sur l'affichage présent à l'écran. Les 4 <modes d'encre> sont les suivants :

0: Normal
1: XOR (OU exclusif)
2: AND (ET)
3: OR (OU)

Mots clés associés : **DRAW, GRAPHICS PEN, MASK**

EDIT

EDIT <numéro de ligne>

```
EDIT 20
```

COMMANDE : Affiche la ligne du programme ainsi que le curseur, prêt à l'édition.

Mots clés associés : **AUTO, LIST**

EI

EI

```
EI
```

COMMANDE : Active (Enables) une Interruption désactivée par **DI**.

Des interruptions désactivées par un sous-programme d'interruption sont automatiquement rétablies par la commande **RETURN**, en fin de sous-programme.

Pour de plus amples informations, consultez la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **AFTER, DI, EVERY, REMAIN**

ELSE

(voir **IF**)

END

END

```
END
```

COMMANDE : Met fin à l'exécution d'un programme et rétablit le mode direct. Un programme peut contenir un nombre quelconque de commandes **END** (elle est implicite à la fin de tout programme BASIC Schneider).

Mots clés associés : **STOP**

ENT

ENT <numéro d'enveloppe> [,<section d'enveloppe>] [,<section d'enveloppe>] [,<section d'enveloppe>] [,<section d'enveloppe>] [,<section d'enveloppe>]

```
10 ENT 1,10,-50,10,10,50,10
20 SOUND 1,500,200,10,,1
run
```

COMMANDE : Définit l'**EN**veloppe de Tonalité spécifiée par le <numéro d'enveloppe> (entre 1 et 15) utilisée avec la commande **SOUND**. Si le <numéro d'enveloppe> est négatif (entre -1 et -15) l'enveloppe se répète jusqu'à la fin de la durée du son fixée par la commande **SOUND**.

Chaque <section d'enveloppe> peut contenir 2 ou 3 paramètres :

Dans le cas de trois paramètres, ceux-ci sont :

<nombre de pas> , <amplitude du pas> , <durée du pas>.

Paramètre 1 : <nombre de pas>

Spécifie le <nombre de pas> de variation de tonalité à l'intérieur de la section d'enveloppe. Par exemple, dans une section de note durant 10 secondes, vous pouvez fixer 10 pas de 1 seconde chacun. Dans ce cas, le <nombre de pas> sera 10.

Le <nombre de pas> peut varier de 0 à 239.

Paramètre 2 : <amplitude du pas>

Doit être compris entre -128 et + 127. Les pas négatifs augmentent la hauteur de la note, les pas positifs l'abaissent. La période sonore minimale est de 0. Toutes les périodes sonores disponibles sont présentées au chapitre « Pour information... ».

Paramètre 3 : <durée du pas>

Spécifie la durée d'un pas, par unités de 0,01 seconde. Elle peut varier de 0 à 255 (0 a la valeur 256), la durée maximale d'un pas est donc de 2,56 secondes. Si vous n'utilisez que deux paramètres, ce sont :

<période sonore>,<durée du pas>

Paramètre 1 : <période sonore>

Donne la nouvelle valeur de la période. (Voir le paramètre 2 de la commande SOUND)

Paramètre 2 : <durée du pas>

Spécifie la durée du pas unique en unités de 0,01 seconde. Peut varier entre 0 et 255 (0 ayant la valeur 256), soit 2,56 secondes.

GÉNÉRALITÉS

La durée totale des pas ne doit pas dépasser le paramètre <durée> de la commande **SOUND**, car le son se termine alors avant d'avoir traversé la totalité des pas. (Le reste de l'enveloppe de tonalité est alors ignoré). De même, si la <durée> de la commande **SOUND** dépasse la durée totale des pas, le son se poursuit après avoir traversé tous les pas et demeure constant à la tonalité finale.

La commande **ENT** peut s'accompagner de 5 <sections d'enveloppe> différentes (chacune constituée des 2 ou 3 paramètres ci-dessus).

Le premier pas d'une enveloppe de tonalité s'exécute immédiatement.

Chaque fois qu'une nouvelle enveloppe est attribuée à un numéro d'enveloppe déterminé, la définition précédente est perdue.

Un <numéro d'enveloppe> sans <section d'enveloppe> annule toutes les spécifications précédentes.

Pour de plus amples informations, consultez la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **ENV**, **SOUND**

ENV

ENV <numéro d'enveloppe> [,<section d'enveloppe>] [,<section d'enveloppe>] [,<section d'enveloppe>] [,<section d'enveloppe>] [,<section d'enveloppe>]

```
10 ENV 1,15,-1,10,15,1,10
20 SOUND 1,200,300,15,1
run
```

COMMANDE : Définit l'**EN**veloppe de **V**olume correspondant au <numéro d'enveloppe> (entre 1 et 15), utilisé avec la commande **SOUND**.

Elle peut contenir 2 ou 3 paramètres :

Pour 3 paramètres :

<nombre de pas>,<amplitude du pas>,<durée du pas>.

Paramètre 1 : <nombre de pas>

Spécifie le <nombre de pas> de volume qu'un son doit traverser dans la section d'enveloppe. Par exemple, dans une section de 10 secondes, vous pouvez fixer 10 pas de volume d'une seconde. Le paramètre <nombre de pas> est égal à 10.

Le paramètre peut varier de 0 à 127.

Paramètre 2 : <amplitude du pas>

Peut faire varier le volume de 0 à 15 par rapport au pas précédent. Les 16 volumes différents sont les mêmes que ceux de la commande **SOUND**. Le paramètre <amplitude du pas> peut cependant varier de -128 à +127, le volume revenant à 0 après avoir atteint 15.

Paramètre 3 : <durée du pas>

Spécifie la durée d'un pas en unités de 0,01 seconde. Peut varier de 0 à 255 (0 à la valeur 256), soit 2,56 secondes.

Pour 2 paramètres :

<enveloppe matérielle>,<période de l'enveloppe>

Paramètre 1 : <enveloppe matérielle>

Spécifie la valeur à envoyer au registre d'enveloppe contenu dans le générateur sonore.

Paramètre 2 : <période de l'enveloppe>

Spécifie la valeur à envoyer aux registres de période d'enveloppe contenus dans le générateur sonore.

L'utilisation d'enveloppes matérielles suppose la connaissance du matériel. Si vous ne l'avez pas, il vaut mieux utiliser une enveloppe logicielle intégrant un paramètre <durée du pas> adéquat.

GÉNÉRALITÉS

La durée totale des pas ne doit pas dépasser le paramètre de <durée> de la commande **SOUND**, le son se termine alors avant d'avoir traversé tous les pas de volume (le reste de l'enveloppe est ignoré).

De même, si le paramètre de <durée> de la commande **SOUND** dépasse la durée totale des pas, le son se poursuit après avoir traversé tous les pas de volume et demeure constant au volume final.

La commande **ENV** peut contenir 5 sections d'enveloppes différentes (constituées des 2 ou 3 paramètres ci-dessus).

Le premier pas d'une enveloppe de volume s'exécute immédiatement.

Chaque fois qu'une nouvelle enveloppe est attribuée à un numéro d'enveloppe déterminé, la définition précédente est perdue.

La spécification d'un <numéro d'enveloppe> sans <section d'enveloppe> annule toutes les valeurs précédentes.

Pour de plus amples informations, consultez la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **ENT**, **SOUND**

EOF

EOF

```
10 OPENIN "ex1.bas"
20 WHILE NOT EOF
30 LINE INPUT #9,a$
40 PRINT a$
50 WEND
60 CLOSEIN
run
```

FONCTION : Pour tester l'état d'un fichier ouvert. Donne - 1 (vrai) si l'on est en fin de fichier (**E**nf **O**f **F**ile) ou si aucun fichier n'est ouvert, sinon donne 0 (faux).

Mots clés associés : **OPENIN, CLOSEIN**

ERASE

ERASE <liste de:<nom de variable>

```
DIM a(100),b$(100)
ERASE a,b$
```

COMMANDE : Quand un tableau n'est plus nécessaire, il peut être effacé (**ERASE**) afin de libérer la mémoire pour d'autres utilisations.

Mots clés associés : **DIM**

ERL

ERL

```
10 ON ERROR GOTO 30
20 GOTO 1000
30 PRINT"L'erreur est en ligne";ERL
40 END
run
```

FONCTION : Donne le numéro de ligne de la dernière erreur rencontrée. Dans l'exemple ci-dessus l'**ER**reur de la **L**igne **20** est indiquée par la fonction **ERL**.

Mots clés associés : **DERR, ERR, ERROR, ON ERROR GOTO, RESUME**

ERR

ERR

```
GOTO 500
Line does not exist
Ready
PRINT ERR
 8
```

FONCTION : Donne le numéro de la dernière ERReur détectée. Voir la liste des messages d'erreur au chapitre « Pour information... ». Dans l'exemple ci-dessus, le numéro d'erreur **8** correspond à « **Line does not exist** » (Ligne non existante).

Mots clés associés : **DERR**, **ERL**, **ERROR**, **ON ERROR GOTO**, **RESUME**

ERROR

ERROR <nombre entier>

```
10 IF INKEY$="" THEN 10 ELSE ERROR 17
run
```

COMMANDE : Décide une action consécutive à une erreur numérotée. Voir la liste des messages d'erreur 1 à 32 au chapitre « Pour information... ». L'action est la même que celle prévue par le BASIC en cas d'erreur réelle, faisant appel à un sous-programme de traitement d'erreur, le cas échéant, et rapportant les valeurs appropriées d'**ERR** et **ERL**.

ERROR accompagné d'un <nombre entier> compris entre 33 et 255 peut servir à créer des messages d'erreur personnalisés tels que :

```
10 ON ERROR GOTO 100
20 INPUT "entrez un caractere";a$
30 IF LEN(a$)<>1 THEN ERROR 100
40 GOTO 20
100 IF ERR=100 THEN 110 ELSE 130
110 PRINT CHR$(7)
120 PRINT"j'ai dit UN caractere !"
130 RESUME 20
run
```

Mots clés associés : **ERL**, **ERR**, **ON ERROR GOTO**, **RESUME**

EVERY

EVERY <période du chronomètre>[,<numéro du chronomètre>] GOSUB <numéro de ligne>

```
10 EVERY 50,1 GOSUB 30
20 GOTO 20
30 SOUND 1,20
40 RETURN
run
```

COMMANDE : Appelle un sous-programme du BASIC à intervalles réguliers. La <période du chronomètre> spécifie l'intervalle par unités de 0,02 seconde. Le <numéro de chronomètre> (compris entre 0 et 3) spécifie lequel des quatre chronomètres utiliser. Le chronomètre 3 correspond à la priorité supérieure et le 0 à la priorité inférieure. Chaque chronomètre peut être associé à un sous-programme.

Pour de plus amples informations sur les interruptions, consultez la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **AFTER, REMAIN**

EXP

EXP (<expression numérique>

```
PRINT EXP(6.876)
 968.743625
```

FONCTION : Calcule « e » à la puissance donnée par l'<expression numérique> où « e » est égal à 2,7182818 environ, le nombre dont le logarithme naturel est 1.

Mots clés associés : **LOG**

FILL

FILL <encre>

```
10 MODE 0
20 FOR n=1 TO 500
30 PRINT "O";
40 NEXT
50 coulstylo=2+RND*13
60 FILL coulstylo
70 GOTO 50
run
```

COMMANDE : Remplit une zone arbitraire de l'écran graphique. Les bords de la zone sont délimités par les lignes dessinées avec l'encre du stylo en cours ou avec l'encre du fond (comprise entre 0 et 15). Le remplissage part de la position du curseur graphique. Si celui-ci se trouve sur un bord, rien n'est rempli.

Mots clés associés : **GRAPHICS PEN**

FIX

FIX (<expression numérique>)

```
PRINT FIX(9.99999)
 9
```

FONCTION : Enlève la partie décimale d'une <expression numérique> en arrondissant toujours par défaut.

Mots clés associés : **CINT**, **INT**, **ROUND**

FN

(voir **DEF FN**)

FOR

FOR <variable simple> = <début> **TO** <fin >[**STEP** <incrément>]

```
10 FOR n=2 TO 8 STEP 2
20 PRINT n;
30 NEXT n
40 PRINT",qui va augmenter !"
run
```

COMMANDE : Exécute la partie du programme se trouvant entre les mots clés **FOR** et **NEXT**, autant de fois que l'on peut ajouter l'<incrément> à la <variable simple> en partant du <début> jusqu'à la <fin>. Si l'<incrément> est omis, il prend implicitement la valeur **1**.

La valeur de l'<incrément> peut être négative. Dans ce cas la valeur du paramètre <début> doit être supérieure à celle du paramètre <fin>, faute de quoi la variable ne peut être incrémentée.

Les boucles **FOR NEXT** peuvent être imbriquées.

L'affectation du nom de variable à la commande NEXT est facultative car le BASIC détermine automatiquement la commande NEXT à laquelle est associée une commande FOR.

Mots clés associés : **NEXT**, **STEP**, **TO**

FRAME

FRAME

```
10 MODE 0
20 PRINT"sans FRAME"
30 TAG
40 MOVE 0,200
50 FOR x=0 TO 500 STEP 4
60 IF f=1 THEN FRAME
70 MOVE x,200
80 PRINT" ";CHR$(143);
90 NEXT
100 IF f=1 THEN RUN
110 CLS
120 TAGOFF
130 PRINT"avec FRAME"
140 f=1
150 GOTO 30
run
```

COMMANDE : Synchronise l'écriture des graphiques avec les trames vidéo. Il en résulte un mouvement plus harmonieux des caractères ou des graphiques à l'écran sans distorsion et sans scintillement.

Mots clés associés : **TAG**, **TAGOFF**

FRE

FRE <expression numérique>)
FRE (<chaîne alphanumérique>)

```
PRINT FRE(0)
PRINT FRE("")
```

FONCTION : Indique l'espace disponible en mémoire. La forme **FRE** (" ") force l'ordinateur à mettre de l'ordre avant de donner la valeur de l'espace disponible.

REMARQUE : Le BASIC n'exploite que le bloc **0** de la mémoire.

Mots clés associés : **HIMEM**, **MEMORY**

GOSUB

GOSUB <numéro de ligne>

```
GOSUB 210
```

COMMANDE : Appelle un sous-programme BASIC en se branchant sur la ligne indiquée. La fin du sous-programme est marquée par la commande **RETURN** renvoyant le programme à l'instruction suivant la commande **GOSUB**.

Mots clés associés : **RETURN**

GOTO

GOTO <numéro de ligne>

```
GOTO 90
```

COMMANDE : Se branche sur le numéro de ligne indiqué.

Mots clés associés : Aucun

GRAPHICS PAPER

GRAPHICS PAPER <encre>

```
10 MODE 0
20 MASK 15
30 GRAPHICS PAPER 3
40 DRAW 640,0
run
```

COMMANDE : Détermine l'<encre> du fond graphique. Lors du traçage de lignes, le fond n'est pas visible. Dans l'exemple ci-dessus, la commande **MASK** permet le traçage d'une ligne en tirets et la visualisation du fond graphique.

L'encre du fond (entre 0 et 15) sert à la zone « paper » sur laquelle s'affichent les caractères, lorsque **TAG** fonctionne, et fait office de valeur par défaut lors de l'effacement des fenêtres graphiques à l'aide de **CLG**.

Mots clés associés : **CLG, GRAPHICS PEN, INK, MASK, TAG, TAGOFF**

GRAPHICS PEN

GRAPHICS PEN [<encre>][,<mode du fond>]

```
10 MODE 0
20 GRAPHICS PEN 15
30 MOVE 200,0
40 DRAW 200,400
50 MOVE 639,0
60 FILL 15
run
```

COMMANDE : Fixe l'<encre> (entre 0 et 15) pour le dessin des lignes et le positionnement des points. Le <mode du fond> peut également être fixé :

0: Fond opaque
1: Fond transparent

(Le fond transparent a une influence sur le fond graphique des caractères écrits avec **TAG** et les espaces en lignes pointillées).

Vous pouvez omettre l'un des paramètres mais pas les deux. Si l'un des paramètres est omis, la valeur spécifiée reste inchangée.

Mots clés associés : **GRAPHICS PAPER**, **INK**, **MASK**, **TAG**, **TAGOFF**

HEX$

HEX$ (<nombre entier sans signe>[,<largeur de zone>])

```
PRINT HEX$(255,4)
00FF
```

FONCTION : Change un nombre entier en un nombre HEXadécimal équivalent en accord avec le nombre de chiffres hexadécimaux indiqué par la <largeur de zone> (entre 0 et 16). Si ce nombre est trop grand, l'expression finale est complétée par des zéros à gauche du nombre. S'il est trop petit, elle ne sera PAS tronquée mais le nombre de chiffres produit sera égal au nombre requis.

Le <nombre entier sans signe> à convertir sous forme hexadécimale doit produire une valeur comprise entre -32768 et 65535.

Mots clés associés : **BIN$**, **DEC$**, **STR$**, **UNT**

HIMEM

HIMEM

```
PRINT HIMEM
42619
```

FONCTION : Donne l'adresse de l'octet le plus **H**aut dans la **MEM**oire utilisée par le BASIC (modifiable par la commande **MEMORY**).

REMARQUE : Le BASIC n'exploite que le bloc **0** de la mémoire.

Mots clés associés : **FRE**, **MEMORY**, **SYMBOL**, **SYMBOL AFTER**

IF

IF <expression logique> **THEN** <option>[**ELSE** <option>]

```
10 MODE 1
20 x=CINT(RND*100)
30 PRINT"devinez un chiffre (0 a 100)"
40 INPUT n
50 IF n<x THEN PRINT n;"est trop petit..."
60 IF n>x THEN PRINT n;"est trop grand..."
70 IF n=x THEN 80 ELSE c=c+1:GOTO 40
80 PRINT"Bien vu! ";"Trouve en";c+1;"fois !"
run
```

COMMANDE : Détermine si l'<expression logique> est vraie pour exécuter, le cas échéant, la première <option>. Si l'<expression logique> est fausse, l'<option> placée après **ELSE** s'exécute. En absence de deuxième <option>, le BASIC passe à la ligne suivante.

Les commandes **IF THEN** peuvent être imbriquées mais se terminent à la fin de la ligne. On ne peut donc PAS avoir de déclarations indépendantes de **IF THEN** sur la même ligne. Lorsque le résultat de l'<expression logique> nécessite un saut de ligne, la commande peut, par exemple, se formuler :

```
IF a=1 THEN 100
```

...ou :

```
IF a=1 GOTO 100
```

...ou :

```
IF a=1 THEN GOTO 100
```

Mots clés associés : **ELSE**, **GOTO**, **THEN**

INK

INK <encre>,<numéro de couleur>[,<numéro de couleur>]

```
10 MODE 1:PAPER 0:PEN 1
20 FOR p=0 TO 1
30 FOR i=0 TO 26
40 INK p,i
50 LOCATE 16,12:PRINT"INK";p;",";i
60 FOR t=1 TO 400:NEXT t,i,p
70 INK 0,1:INK 1,24:CLS
run
```

COMMANDE : Affecte la ou les couleurs à une encre donnée. Le paramètre <encre> fournit la référence de l'encre (par un entier compris entre 0 et 15), à l'intention des commandes **PEN** ou **PAPER** correspondantes. Le premier paramètre <numéro de couleur> (entier) donne une valeur de couleur comprise entre 0 et 26. Si le second paramètre de couleur facultatif est spécifié, l'encre passe d'une couleur à l'autre selon une vitesse définie par la commande **SPEED INK**.

Mots clés associés : **GRAPHICS PAPER**, **GRAPHICS PEN**, **PAPER**, **PEN**, **SPEED INK**

INKEY

INKEY (<nombre entier>)

```
10 IF INKEY(55)<>32 THEN 10
20 PRINT"vous venez d'appuyer [SHIFT] et V"
30 CLEAR INPUT
run
```

FONCTION : Interroge le clavier pour indiquer les touches pressées. Le clavier est analysé tous les cinquantièmes de seconde. Cette fonction sert à détecter la position haute ou basse d'une touche par détection de la valeur -1 (indépendante de l'état des touches [**SHIFT**] et [**CONTROL**]).

Dans l'exemple ci-dessus, le système détecte l'actionnement simultané de **[SHIFT]** et **V** (numéro de touche 55) avant d'arrêter le programme. Les numéros de touche sont donnés dans le diagramme situé en haut à droite du boîtier de l'ordinateur (voir également le chapitre « Pour information... ».

[SHIFT] et **[CONTROL]** sont identifiés par les valeurs suivantes :

Valeur	**[SHIFT]**	**[CONTROL]**	touche spécifiée
-1	INDIFFERENT	INDIFFERENT	RELEVEE
0	RELEVEE	RELEVEE	ENFONCEE
32	ENFONCEE	RELEVEE	ENFONCEE
128	RELEVEE	ENFONCEE	ENFONCEE
160	ENFONCEE	ENFONCEE	ENFONCEE

Mots clés associés : **CLEAR INPUT**, **INKEY$**, **JOY**

INKEY$

INKEY$

```
10 CLS
20 PRINT"choisissez OUI ou NON (O/N)"
30 a$=INKEY$
40 IF a$="" THEN 30
50 IF a$="o"OR a$="O" THEN 80
60 IF a$="n"OR a$="N" THEN 90
70 GOTO 30
80 PRINT"Vous avez choisi OUI":END
90 PRINT"Vous avez choisi NON"
run
```

FONCTION : Interroge le clavier pour introduire dans le programme toute chaîne de caractères entrée. Si aucune touche de clavier n'est actionnée, **INKEY$** renvoie une chaîne vide. Dans l'exemple ci-dessus les lignes **40** et **70** commandent au programme de revenir à la ligne **30** après interrogation du clavier par la fonction **INKEY$**.

Mots clés associés : **CLEAR INPUT**, **INKEY**

INP

INP (<numéro du port>)

```
PRINT INP(&FF77)
 255
```

FONCTION : Lit la valeur contenue dans un port d'entrées-sorties dont l'adresse est transmise par l'argument de cette fonction.

Mots clés associés : **OUT**, **WAIT**

INPUT

INPUT [# <numéro de canal>,][;][<chaîne> <séparateur>]
<liste de:<variable>

```
10 MODE 1
20 INPUT "donnez-moi deux nombres a multiplier (separes p
ar une virgule)";a,b
30 PRINT a;"fois";b;"font";a*b
40 GOTO 20
run
```

COMMANDE : Reçoit les données en provenance du canal précisé (canal # **Ø** en l'absence de spécification).

Un point-virgule après **INPUT** supprime le passage à la ligne après exécution de la commande.

Le <séparateur> peut être un point-virgule ou une virgule. Un point-virgule placé après la chaîne fait apparaître un point d'interrogation ; une virgule le supprime.

Si une entrée erronée est effectuée, un O pour un **Ø** par exemple, BASIC répond :

```
?Redo from start
```

...ou tout autre message d'erreur programmé par vos soins.

Toute réponse au clavier doit se terminer par [**RETURN**].

Mots clés associés : **LINE INPUT**

INSTR

INSTR ([<position de départ>,]<chaîne contenante>,<chaîne contenue>)

```
10 CLS:FOR n=1 TO 26
20 alphabet$=alphabet$+CHR$(n+64)
30 NEXT
40 INPUT "entrez une lettre";a$
50 b$=UPPER$(a$)
60 PRINT b$;" est en position";
70 PRINT INSTR(alphabet$,b$);
80 PRINT"dans l'alphabet.":PRINT
90 GOTO 40
run
```

FONCTION : Cherche dans la <chaîne contenante> l'occurence de la <chaîne contenue> et indique la position de la première occurence de la chaîne recherchée. En son absence, la fonction indique la valeur **Ø**.

La position du début de la recherche est facultative, elle est spécifiée par le paramètre <position de départ> sous la forme d'un entier compris entre 1 et 255.

En cas de recherche infructueuse, la fonction retourne la valeur Ø.

Mots clés associés : Aucun

INT

INT (<expression numérique>)

```
PRINT INT(-1.995)
-2
```

FONCTION : Arrondit au premier entier inférieur, enlevant la partie fractionnaire. Identique à **FIX** pour les nombres positifs, il donne 1 de moins que **FIX** pour les nombres négatifs qui ne sont pas des entiers.

Mots clés associés : **CINT**, **FIX**, **ROUND**

JOY

JOY (<nombre entier>)

```
10 PRINT"Pour arreter le programme - ";
20 PRINT" actionnez la manette de jeu"
30 IF JOY(0)<>0 THEN END
40 GOTO 10
run
```

FONCTION : La fonction **JOY** lit l'état de la manette de jeu spécifiée par le <nombre entier> (**Ø** ou **1**). Le résultat n'a de signification qu'en binaire.

Bit	Décimal
0: Haut	1
1: Bas	2
2: Gauche	4
3: Droite	8
4: Tir 2	16
5: Tir 1	32

Ainsi, lorsque vous appuyez sur le bouton de « tir » (Tir 2) de la première manette en déplaçant celle-ci vers la gauche, la fonction **JOY (Ø)** envoie une valeur décimale égale à 20, soit 16 (Tir 2) + 4 (Gauche).

Pour de plus amples informations, consultez le chapitre « Pour information... ».

Mots clés associés : **CLEAR INPUT, INKEY**

KEY

KEY <numéro logique de touche>,<chaîne alphanumérique>

```
KEY 11,"border 13:paper 0:pen 1:ink 0,13:ink 1,0:mode 2:
list "+CHR$(13)
```

Appuyez sur la touche **[ENTER]**.

COMMANDE : Associe une chaîne à la touche (**KEY**) correspondant au <numéro logique de touche> spécifiée. Il existe 32 numéros logiques de touche (de 0 à 31), occupant les touches 128 à 159. Les touches 128 (**Ø** sur le clavier numérique) à 140 (**[CONTROL] [ENTER]** sur le clavier numérique) sont associées par défaut aux chiffres **Ø** à **9**, au point décimal, à **[RETURN]** et à RUN" **[RETURN]** - (pour la cassette), mais peuvent être associées à d'autres chaînes si nécessaire. Les numéros logiques de touche 13 à 31 (touches 141 à 159) sont affectés à des chaînes vides par défaut mais peuvent être étendus et associés à des touches, à l'aide de la commande **KEY DEF**.

Le <numéro logique de touche> fourni dans la commande **KEY** doit être compris entre 0 et 31 ou entre 128 et 159 pour correspondre aux numéros physiques des touches du clavier numérique. (Voir la représentation des touches au chapitre « Pour information... »).

La chaîne spécifiée ne doit pas dépasser 120 caractères au total . Le dépassement de cette limite entraîne une erreur (5) « **Improper argument** » (argument incorrect).

Mots clés associés : **KEY DEF**

KEY DEF

KEY DEF <numéro de touche>,<répétition>[,<normal>[,<avec shift> [,<avec control>]]]

```
KEY 159,"c'est la touche TAB"
KEY DEF 68,1,159
```

Appuyez sur la petite touche **[TAB]**

COMMANDE : Définit la valeur logique d'une touche (**KEY**) DEFinie par son numéro physique, compris entre 0 et 79 (voir l'illustration des numéros de touche en haut à droite de l'ordinateur ou au chapitre « Pour information... »). Les paramètres <normal>, <avec shift> et <avec control> doivent contenir les valeurs à envoyer lorsque la touche est enfoncée, seule, avec [**SHIFT**] ou avec [**CONTROL**]. Ces paramètres sont facultatifs.

Le paramètre <répétition> permet d'activer et de désactiver la fonction d'auto-répétition (**1** ou **Ø**). La vitesse de celle-ci est réglable par la commande **SPEED KEY**.

Dans l'exemple ci-dessus, la touche 159 (correspondant au numéro logique 31) est d'abord associée à une chaîne. La commande **KEY DEF** définit ensuite la touche **68** (touche **[TAB]**) pour activer l'auto-répétition (**1**) et envoyer la valeur <normal> **159** lorsque la touche est enfoncée.

Pour revenir au mode normal :

```
KEY DEF 68,0,9
```

...**9** étant la valeur ASCII normale de **[TAB]**.

Mots clés associés : **KEY**, **SPEED KEY**

LEFT$

LEFT$ (<chaîne alphanumérique>,<longueur requise>)

```
10 CLS
20 a$="AMSTRAD"
30 FOR n=1 TO 7
40 PRINT LEFT$(a$,n)
50 NEXT
run
```

FONCTION : Extrait un certain nombre de caractères (entre 0 et 255) à gauche d'une <chaîne alphanumérique>. Si la chaîne est plus courte que la longueur requise, elle est utilisée entièrement.

Mots clés associés : **MID$**, **RIGHT$**

LEN

LEN (<chaîne alphanumérique>)

```
10 LINE INPUT "entrez une phrase";a$
20 PRINT"la phrase est longue de";
30 PRINT LEN(a$);"caracteres."
run
```

FONCTION : Donne le nombre de caractères de la <chaîne alphanumérique>, les espaces compris.

Mots clés associés : Aucun

LET

LET <variable> = <expression>

```
LET x=100
```

COMMANDE : Un reste des BASIC historiques, pour lesquels on devait annoncer ses variables. Seulement utile pour la compatibilité avec des programmes antérieurs. En Schneider BASIC, il suffit d'écrire :

```
x=100
```

Mots clés associés : Aucun

LINE INPUT

LINE INPUT # [<numéro de canal>,][;][<chaîne> <séparateur>] <variable en chaîne>

```
10 LINE INPUT "tapez une ligne de texte ponctuee.";a$
20 CLS
30 PRINT"la variable a$ est bien egale a :-"
40 PRINT a$
run
```

COMMANDE : Reçoit une ligne entière en provenance du canal indiqué (# **0** en l'absence de spécification). Le premier **[;]** point-virgule facultatif supprime le retour chariot/saut de ligne qui intervient normalement après exécution de la commande.

Le <séparateur> peut être un point-virgule ou une virgule. Le point-virgule entraîne l'affichage d'un point d'interrogation ; la virgule le supprime.

L'entrée de **LINE INPUT** au clavier se termine par l'activation de la touche **[RETURN]**.

LINE INPUT en provenance du canal # **9** de la disquette (ou de la cassette) se termine par un retour chariot ou par l'affectation de plus de 255 caractères à la <variable en chaîne>.

Mots clés associés : **INPUT**

LIST

LIST [<ensemble de lignes>][, <numéro de canal>]

```
LIST 100-1000,#1
```

COMMANDE : Liste le programme sur le canal désiré. # **0** est l'écran, # **8** est l'imprimante. Le **LIST**age peut être provisoirement interrompu si vous appuyez une fois sur la touche **[ESC]**, pour être ensuite repris à l'aide de la barre d'espacement. Si vous appuyez deux fois sur **[ESC]**, vous arrêtez le listage et revenez au mode direct.

Vous pouvez omettre le premier ou le dernier numéro de ligne du paramètre <ensemble de lignes> pour lister le programme depuis le début, ou jusqu'à la fin.

Exemples :

```
LIST -200
```

...ou :

```
LIST 50-
```

...ou :

```
LIST
```

...liste la totalité du programme.

Mots clés associés : Aucun

LOAD

LOAD <nomfich>[,<adresse>]

```
LOAD "fichdisc.xyz",&2AF8
```

COMMANDE : Charge en mémoire un programme BASIC se trouvant sur disquette, en écrasant tout programme en place. Avec l'option de l'adresse, charge un fichier binaire à l'adresse indiquée au lieu de l'adresse à laquelle il se trouvait au moment de la sauvegarde.

Un programme protégé ne peut PAS être chargé par la commande **LOAD** car il est alors immédiatement effacé de la mémoire. Dans ce cas, utilisez **RUN** ou **CHAIN**.

Mots clés associés : **CHAIN**, **CHAIN MERGE**, **MERGE**, **RUN**, **SAVE**

LOCATE

LOCATE[#<numéro de canal>,]<coordonnée x>,<coordonnée y>

```
10 MODE 1
20 FOR n=1 TO 20
30 LOCATE n,n
40 PRINT CHR$(143);"position";
50 PRINT n;",";n
60 NEXT
run
```

COMMANDE : Déplace le curseur de texte vers une nouvelle position relative au coin supérieur gauche de la fenêtre (WINDOW). # **Ø** représente le canal par défaut.

Mots clés associés : **WINDOW**

LOG

LOG (<expression numérique>)

```
PRINT LOG(9999)
 9.21024037
```

FONCTION : Calcule le **LOG**arithme naturel d'une expression numérique (supérieure à 0).

Mots clés associés : **EXP**, **LOG 1Ø**

LOG 1Ø

LOG 1Ø <expression numérique>)

```
PRINT LOG10(9999)
 3.99995657
```

FONCTION : Calcule le **LOG**arithme à base **1Ø** de l'<expression numérique> (supérieure à zéro).

Mots clés associés : **EXP**, **LOG**

LOWER$

LOWER$ (<chaîne alphanumérique>)

```
10 a$="REGARDEZ COMMENT LES LETTRES SONT CHANGEES "
20 PRINT LOWER$(a$+"EN TYPE MINUSCULE")
run
```

FONCTION : Change toutes les majuscules d'une chaîne alphanumérique en minuscules. Utile quand on attend des réponses composées d'un mélange de majuscules et de minuscules.

Mots clés associés : **UPPER$**

MASK

MASK [<nombre entier>][,<tracé du premier point>]

```
10 MODE 0:INK 5,21:INK 8,16
20 MOVE -100*RND,400*RND
30 WHILE XPOS<640
40 FOR x=1 TO 8
50 MASK 2↑(8-x)
60 DRAWR 32,0,x,1:MOVER -32,0
70 NEXT
80 MOVER 34,0
90 WEND:GOTO 20
run
```

COMMANDE : Définit le modèle à utiliser pour le tracé des lignes. La valeur binaire du <nombre entier> comprise entre 0 et 255 active (1) ou désactive (0) les bits dans chaque groupe adjacent de 8 pixels.

Le paramètre <tracé du premier point> détermine si le premier point de la ligne doit être tracé (1) ou non (0).

Vous pouvez omettre l'un des paramètres mais pas les deux. Si vous en omettez un, sa spécification demeure inchangée.

Mots clés associés : **DRAW**, **DRAWR**, **GRAPHICS PAPER**, **GRAPHICS PEN**

MAX

MAX (<liste de:<expression numérique>)

```
10 n=66
20 PRINT MAX(1,n,3,6,4,3)
run
 66
```

FONCTION : Donne la valeur la plus grande (**MAX**imale) de la liste.

Mots clés associés : **MIN**

MEMORY

MEMORY <adresse>

```
MEMORY &20AA
```

COMMANDE : Définit la quantité de mémoire BASIC disponible en fixant l'adresse de l'octet le plus élevé.

REMARQUE : Le BASIC n'exploite que le bloc **O** de la mémoire.

Mots clés associés : **FRE**, **HIMEM**, **SYMBOL**, **SYMBOL AFTER**

MERGE

MERGE <nomfich>

```
MERGE "nouversi.bas"
```

COMMANDE : Charge un programme à partir de la disquette et l'ajoute au programme déjà en mémoire.

Les numéros de ligne du programme en place se retrouvant dans le nouveau programme sont automatiquement écrasés.

Les fichiers protégés (**SA**u**VE**gardés par **SAVE,p**) ne peuvent PAS être fusionnés avec un autre programme en place.

Mots clés associés : **CHAIN**, **CHAIN MERGE**, **LOAD**

MID$

MID$ (<chaîne alphanumérique>,<position de départ>[,<longueur de la sous-chaîne>])

```
10 MODE 1:ZONE 3
20 a$="ENCYCLOPEDIE"
30 PRINT"Regardez comment epeler ";a$
40 PRINT
50 FOR n=1 TO LEN(a$)
60 PRINT MID$(a$,n,1),
70 FOR t=1 TO 700:NEXT t,n
80 PRINT:PRINT
90 INPUT "entrez un nouveau mot";a$
100 GOTO 50
run
```

FONCTION : Renvoie une nouvelle sous-chaîne commençant à la <position de départ> de la <chaîne alphanumérique> et contenant le nombre de caractères correspondant à la <longueur de la sous-chaîne>. Si le paramètre <longueur de la sous-chaîne> n'est pas spécifié, la fonction renvoie le reste de la <chaîne alphanumérique> à partir de la <position de départ>.

Si la <position de départ> est supérieure à la longueur totale de la <chaîne alphanumérique>, une chaîne vide est renvoyée. La <position de départ> est comprise entre 1 et 255, la <longueur de la sous-chaîne> entre 0 et 255.

Mots clés associés : **LEFT$**, **RIGHT$**

MID$

MID$ (<variable chaîne>,<position d'insertion>[,<longueur de la nouvelle chaîne>]) = <nouvelle chaîne alphanumérique>

```
10 a$="bonjour"
20 MID$(a$,3,2)="XX"
30 PRINT a$
run
boXXour
```

COMMANDE/ Insère dans la chaîne spécifiée une <nouvelle chaîne alphanumérique> d'un nombre de caractères donné, à la <position d'insertion>.

Lorsque vous utilisez **MID$** en tant que COMMANDE, vous devez faire appel à une <variable chaîne>, tel que **A$**, et non PAS à une constante comme « **bonjour** ».

Mots clés associés : **LEFT$**, **RIGHT$**

MIN

MIN (<liste de:<expression numérique>)

```
PRINT MIN(3,6,2.999,8,9,)
2.999
```

FONCTION : Donne la valeur la plus petite (**MIN**imale) de la <liste de:<expressions numériques>.

Mots clés associés : **MAX**

MOD

<argument 1> MOD <argument 2>

```
PRINT 10 MOD 3
 1
PRINT 10 MOD 5
 0
```

OPERATEUR : Donne le reste de la division entière de l'<argument 1> par l'<argument 2> (on dit <argument 1> **MOD**ulo <argument 2>).

Mots clés associés : Aucun

MODE

MODE <nombre entier>

```
10 m=m+1:IF m>2 THEN m=0
20 MODE m
30 PRINT"Ceci est le mode";m
40 PRINT"Pressez une touche"
50 IF INKEY$="" THEN GOTO 50 ELSE 10
run
```

COMMANDE : Modifie le mode d'écran (0, 1 ou 2) et rétablit sur l'écran l'encre 0, même si l'encre actuellement utilisée par le papier est différente. Toutes les fenêtres et curseurs sont réinitialisés.

Mots clés associés : **WINDOW**, **ORIGIN**

MOVE

MOVE <coordonnée x>,<coordonnée y>[,[<encre>][,<mode d'encre>]]

```
10 MODE 1:TAG
20 x=RND*800-100:y=RND*430
30 MOVE x,y
40 PRINT"je suis ici";
50 GOTO 20
run
```

COMMANDE : Positionne le curseur graphique au point absolu spécifié. Le paramètre facultatif <encre> (compris entre 0 et 15) permet de changer la couleur du stylo graphique.

Le paramètre facultatif <mode d'encre> détermine l'interaction de l'encre sur l'affichage en place à l'écran. Il existe 4 <modes d'encre> :

0: Normal
1: XOR (OU exclusif)
2: AND (ET)
3: OR (OU)

Mots clés associés : **MOVER**, **ORIGIN**, **XPOS**, **YPOS**

MOVER

MOVER <décalage x>,<décalage y>[,[<encre>][,<,mode d'encre>]]

```
10 MODE 1:TAG:MOVE 0,16
20 PRINT"la vie a ses";
30 FOR n=1 TO 10
40 MOVER -45,16
50 PRINT"hauts";:NEXT:PRINT" et";
60 FOR n=1 TO 10
70 MOVER -64,-16
80 PRINT"bas";:NEXT
run
```

COMMANDE : Positionne le curseur graphique en coordonnées relatives (par rapport à la position actuelle). Le paramètre facultatif <encre> (compris entre 0 et 15) permet de changer la couleur du stylo graphique.

Le paramètre facultatif <mode d'encre> détermine l'interaction de l'encre sur l'affichage en place à l'écran. Il existe 4 <modes d'encre> :

0: Normal
1: XOR (OU exclusif)
2: AND (ET)
3: OR (OU)

Mots clés associés : **MOVE**, **ORIGIN**, **XPOS**, **YPOS**

NEW

NEW

```
NEW
```

COMMANDE : Efface le programme et les variables en mémoire. Les définitions de touches ne sont pas effacées et l'affichage reste ce qu'il était (**MODE**, **PEN**, **PAPER**, **INK** etc.).

Mots clés associés : Aucun

NEXT

NEXT [<liste de:<variable>]

```
10 FOR a=1 TO 3
20 FOR b=0 TO 26
30 MODE 1
40 PEN a:BORDER b
50 PRINT"PEN";a;"BORDER";b
60 FOR c=1 TO 500
70 NEXT c,b,a
run
```

COMMANDE : Marque la fin d'une boucle commencée avec **FOR**. La commande **NEXT** peut être anonyme ou peut se rapporter au **FOR** concerné. Dans l'exemple ci-dessus, la <liste de:variable> doit apparaître en sens inverse de la spécification des commandes **FOR**, afin d'éviter le chevauchement des boucles imbriquées.

Mots clés associés : **FOR**, **STEP**, **TO**

NOT

NOT <argument>

```
IF NOT "alain" < "bernard" THEN PRINT "vrai" ELSE  PRINT
"faux"
faux
IF NOT "chat" > "chien" THEN  PRINT  "vrai"  ELSE  PRINT
"faux"
vrai
....
PRINT NOT -1
 0
PRINT NOT 0
-1
```

OPERATEUR : Exécute des opérations par bit sur des entiers. Inverse chaque bit de l'argument. Pour de plus amples informations, consultez la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **AND**, **OR**, **XOR**

ON BREAK CONT

ON BREAK CONT

```
10 ON BREAK CONT
20 PRINT"Le programme CONTinuera si vous essayez de faire *B
reak* avec la touche [ESC]":PRINT
30 FOR t=1 TO 1000:NEXT:GOTO 20
run
```

COMMANDE : Annule l'action de la touche **[ESC]**, empêchant l'arrêt du programme. Cette commande est à utiliser avec précaution car le programme ne peut alors être interrompu que par réinitialisation complète de l'ordinateur (vous devez donc sauvegarder le programme avant de le lancer).

Vous pouvez désactiver **ON BREAK CONT** par **ON BREAK STOP** à l'intérieur d'un programme.

Mots clés associés : **ON BREAK GOSUB, ON BREAK STOP**

ON BREAK GOSUB

ON BREAK GOSUB <numéro de ligne>

```
10 ON BREAK GOSUB 40
20 PRINT"le programme tourne"
30 GOTO 20
40 CLS:PRINT"Appuyer 2 fois [ESC],";
50 PRINT"appelle le sous-programme"
60 FOR t=1 TO 2000:NEXT
70 RETURN
run
```

COMMANDE : Demande au BASIC de passer au sous-programme spécifié par le <numéro de ligne> lorsque vous appuyez deux fois sur **[ESC]**.

Mots clés associés : **ON BREAK CONT, ON BREAK STOP, RETURN**

ON BREAK STOP

ON BREAK STOP

```
10 ON BREAK GOSUB 40
20 PRINT"le programme tourne"
30 GOTO 20
40 CLS:PRINT"Appuyer 2 fois [ESC],";
50 PRINT"appelle le sous-programme"
60 FOR t=1 TO 2000:NEXT
65 ON BREAK STOP
70 RETURN
run
```

COMMANDE : Désactive les commandes **ON BREAK CONT** et **ON BREAK GOSUB** pour permettre l'arrêt du programme sur activation de la touche **[ESC]**. Dans l'exemple ci-dessus, la commande **ON BREAK GOSUB** ne fonctionne qu'une seule fois car elle est désactivée à la ligne **65** dans le sous-programme **ON BREAK**.

Mots clés associés : **ON BREAK CONT**, **ON BREAK GOSUB**

ON ERROR GOTO

ON ERROR GOTO <numéro de ligne>

```
10 ON ERROR GOTO 60
20 CLS:PRINT"Si une erreur est trouvee, ";
30 PRINT"alors LISTer le programme"
40 FOR t=1 TO 4000:NEXT
50 GOTO 100
60 PRINT"Erreur detectee a la ligne";
70 PRINT ERL:PRINT:LIST
run
```

COMMANDE : Passe à la ligne spécifiée aussitôt qu'une erreur est détectée.

La commande **ON ERROR GOTO 0** désactive le déroutement du programme sur erreur et rétablit le traitement normal des erreurs par le BASIC.

Voir également la commande **RESUME**

Mots clés associés : **DERR**, **ERL**, **ERR**, **ERROR**, **RESUME**

ON <expression> GOSUB

ON <sélecteur> **GOSUB** <liste de:<numéro de ligne>

```
10 PAPER 0:PEN 1:INK 0,1
20 CLS:PRINT" MENU ":PRINT
30 PRINT"1 - Changer le cadre":PRINT
40 PRINT"2 - Changer le stylo":PRINT
50 PRINT"3 - Changer de mode":PRINT
60 INPUT "Votre choix";x
70 ON x GOSUB 90,110,130
80 GOTO 20
90 b=b-1:IF b<0 THEN b=26
100 BORDER b:RETURN
110 p=p-1:IF p<2 THEN p=26
120 INK 1,p:RETURN
130 m=m-1:IF m<0 THEN m=2
140 MODE m:RETURN
run
```

COMMANDE : Sélectionne une ligne de sous-programme en fonction de la valeur du <sélecteur> (nombre entier compris entre 0 et 255). L'ordre des valeurs des <sélecteurs> détermine le numéro de ligne à extraire de la <liste de:<numéros de lignes>. Dans l'exemple ci-dessus :

1 provoque le passage à la ligne **90**,
2 provoque le passage à la ligne **110**,
3 provoque le passage à la ligne **130**.

Si cette expression est égale à zéro, ou si elle est supérieure au nombre de lignes de la liste spécifiée dans la commande, la sélection n'a pas lieu.

Mots clés associés : **RETURN**

ON <expression> GOTO

ON <sélecteur> GOTO <liste de:<numéro de ligne>

```
10  CLS:PRINT" MENU ":PRINT
20 PRINT"1 - LISTe le programme":PRINT
30 PRINT"2 - EDITe pour corriger":PRINT
40 PRINT"3 - Fait le CATalogue":PRINT
50 INPUT "Votre choix":n
60 ON n GOTO 80,90,100
70 GOTO 10
80 LIST
90 AUTO
100 CAT
run
```

COMMANDE : Sélectionne une ligne à laquelle le programme doit sauter en fonction de la valeur du <sélecteur> (nombre entier compris entre 0 et 255). L'ordre des valeurs du <sélecteur> détermine le numéro de ligne à extraire de la <liste de:<numéros de lignes>. Dans l'exemple ci-dessus :

1 provoque le passage à la ligne **80**,
2 provoque le passage à la ligne **90**,
3 provoque le passage à la ligne **100**.

Si cette expression est égale à zéro, ou si elle est supérieure au nombre de lignes de la liste spécifiée dans la commande, la sélection n'a pas lieu.

Mots clés associés : Aucun

ON SQ GOSUB

ON SQ (<numéro de canal>) **GOSUB** <numéro de ligne>

```
10 ENV 1,15,-1,1
20 ON SQ(1) GOSUB 60
30 MODE 0:ORIGIN 0,0,200,440,100,300
40 FOR x=1 TO 13:FRAME:MOVE 330,200,x
50 FILL x:NEXT:GOTO 40
60 READ s:IF s=0 THEN RESTORE:GOTO 60
70 SOUND 1,s,25,15,1
80 ON SQ(1) GOSUB 60:RETURN
90 DATA 50,60,90,100,35,200,24,500,0
run
```

COMMANDE : Provoque un déroutement en cas de place dans une file sonore (Sound Queue). Le <numéro de canal> est un nombre entier indiquant une des valeurs :

1: pour canal A
2: pour canal B
4: pour canal C

Pour de plus amples informations concernant la logique sonore, consultez la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **RETURN**, **SOUND**, **SQ**

OPENIN

OPENIN <nomfich>

```
10 REM Ouvre et Recoit le fichier en provenance de la
disquette
20 OPENIN "NOMFICH":INPUT #9,a,a$
30 CLOSEIN:PRINT"les 2 valeurs sont:"
40 PRINT:PRINT a,a$
run
```

COMMANDE : Ouvre un fichier existant sur la disquette afin d'y lire des données destinées au programme en mémoire. Le fichier à ouvrir doit être un fichier ASCII. Cet exemple ne fonctionne que si vous avez créé le fichier selon l'exemple de la commande OPENOUT.

Mots clés associés : **CLOSEIN**, **EOF**

OPENOUT

OPENOUT <nomfich>

```
10 REM Ouvre et Sort le fichier sur la disquette
20 INPUT "Donnez moi un nombre";a
30 INPUT "Donnez moi un mot";a$
40 OPENOUT "NOMFICH"
50 WRITE #9,a,a$
60 CLOSEOUT:PRINT"les donnees sont sauvees sur disquette"
run
```

COMMANDE : Ouvre sur la disquette un fichier de sortie utilisable par le programme en mémoire.

Mots clés associés : **CLOSEOUT**

OR

<argument> OR <argument>

```
IF "alain" < "bernard" OR "chien" > "chat"  THEN  PRINT
 "vrai" ELSE PRINT "faux"
vrai
IF "bernard" < "alain" OR "chat" > "chien"  THEN  PRINT
 "vrai" ELSE PRINT "faux"
faux
IF "alain" < "bernard" OR "chat" > "chien"  THEN  PRINT
 "vrai" ELSE PRINT "faux"
vrai
....
PRINT 1 AND 1
1
PRINT 0 AND 0
0
PRINT 1 AND 0
1
```

OPERATEUR : Exécute en langage machine des opérations booléennes sur des entiers. Donne 1 dans tous les cas sauf si les deux arguments sont égaux à 0. Pour de plus amples informations sur la logique, consultez la partie 2 du chapitre « A vos heures de loisir... ».

Mots clés associés : **AND**, **NOT**, **XOR**

ORIGIN

ORIGIN <x>,<y>[,<gauche>,<droite>,<haut>,<bas>]

```
10 MODE 1:BORDER 13:TAG
20 ORIGIN 0,0,100,540,300,100
30 GRAPHICS PAPER 3:CLG
40 FOR x=550 TO -340 STEP -10
50 MOVE x,206
60 PRINT"Voici une fenetre graphique ";
70 FRAME:NEXT:GOTO 40
run
```

COMMANDE : Etablit le point d'origine du curseur graphique aux coordonnées <x>, <y> spécifiées. Vous pouvez également fixer les dimensions de la fenêtre graphique par spécification des quatre derniers paramètres facultatifs. Si les coordonnées spécifiées pour la fenêtre graphique se trouvent en dehors de l'écran, les bords de l'écran sont alors considérés comme les limites de la fenêtre.

Mots clés associés : **CLG**

OUT

OUT <numéro du port>,<nombre entier>

```
OUT &F8F4,&FF
```

COMMANDE : Envoie la valeur du <nombre entier> (entre 0 et 255) vers le port de sortie précisé par son adresse.

A déconseiller aux utilisateurs peu avertis.

Mots clés associés : **INP**, **WAIT**

PAPER

PAPER [# <numéro du canal>,]<encre>

```
10 MODE 0:PEN 0:INK 0,13
20 FOR p=1 TO 15
30 PAPER p:CLS
40 LOCATE 7,12:PRINT"PAPER";p
50 FOR t=1 TO 500:NEXT t,p
run
```

COMMANDE : Etablit la couleur du fond pour les caractères. Lors de l'affichage des caractères sur l'écran, sa matrice (la grille) est remplie par l'<encre> correspondant au papier (**PAPER INK**) avant qu'il ne soit lui-même affiché (sauf en cas de mode transparent).

Si le <numéro de canal> n'est pas spécifié, le canal # **0** est pris par défaut.

Le nombre de couleurs disponibles dépend du mode choisi.

Mots clés associés : **INK**, **GRAPHICS PAPER**, **PEN**

PEEK

PEEK (<adresse>)

```
10 MODE 1:ZONE 7
20 WINDOW 1,40,1,2:WINDOW #1,1,40,3,25
30 PRINT"Adresse memoire"
40 LOCATE 20,1:PRINT"Contenu memoire"
50 FOR n=0 TO 65535
60 p=PEEK(n)
70 PRINT #1,n,"(&";HEX$(n);")";
80 PRINT #1,TAB(20);p,"(&";HEX$(p);")"
90 NEXT
run
```

FONCTION : Lit le contenu de la case mémoire Z80 dont l'<adresse> est indiquée entre les parenthèses. Cette adresse doit être comprise entre &0000 et &FFFF (0 et 65535). PEEK n'opère que sur la mémoire vive (RAM), jamais sur la mémoire morte (ROM), et fournit des valeurs comprises entre &00 et &FF (0 et 255).

Mots clés associés : **POKE**

PEN

PEN # [<numéro de canal>,][<encre>][,<mode du fond>]

```
10 MODE 0:PAPER 0:INK 0,13
20 FOR p=1 TO 15
30 PEN p:PRINT SPACE$(47);"PEN";p
40 FOR t=1 TO 500:NEXT t,p:GOTO 20
run
```

COMMANDE : Sélectionne l'<encre> à utiliser (de 0 à 15) pour écrire sur le canal indiqué (par défaut : #0). Le paramètre <mode du fond> peut être soit transparent (1), soit opaque (0).

Au moins un des deux derniers paramètres doit figurer. Si l'un d'eux est omis, sa valeur antérieure reste inchangée.

Mots clés associés : **PAPER**

PI

PI

```
PRINT PI
 3.14159265
```

FONCTION : Fournit la valeur du rapport circonférence-diamètre d'un cercle.

Mots clés associés : **DEG, RAD**

PLOT

PLOT <coordonnée x>,<coordonnée y>[,[<encre>][,<mode d'encre>]]

```
10 MODE 1:BORDER 0:PAPER 0:PEN 1
20 INK 0,0:INK 1,26:INK 2,13,26:DEG
30 FOR x=1 TO 360:ORIGIN 320,200
40 DRAW 50*COS(x),50*SIN(x),1
50 PLOT 100*COS(x),25*SIN(x):NEXT
60 ORIGIN 0,0:t=TIME+700:WHILE TIME<t
70 PLOT RND*640,RND*400:WEND
80 PLOT RND*640,RND*400,2
90 GOTO 90
run
```

COMMANDE : Affiche, en mode graphique, le point de coordonnées x et y. On définit l'<encre> de ce point sur une échelle de 0 à 15.

Le paramètre facultatif <mode d'encre> détermine le mode d'interaction entre la couleur utilisée et celle de l'écran. Voici les quatre modes possibles :

0 : Normal
1 : XOR (OU exclusif)
2 : AND (ET) : OR (OU)

Mots clés associés : **GRAPHICS PEN, PLOTR**

PLOTR

PLOTR <décalage x>,<décalage y>[,[<encre>][,<mode d'encre>]]

```
10 REM utilisez le pave curseur pour dessiner
20 BORDER 0:GRAPHICS PEN 1
30 MODE 1:PLOT 320,200
40 IF INKEY(0)=0 THEN PLOTR 0,1
50 IF INKEY(1)=0 THEN PLOTR 1,0
60 IF INKEY(2)=0 THEN PLOTR 0,-1
70 IF INKEY(8)=0 THEN PLOTR -1,0
80 IF INKEY(9)=0 THEN 30:REM [COPY]=CLS
90 GOTO 40
run
```

COMMANDE : En mode graphique, affiche à l'écran le point de coordonnées x et y relatives à la position du curseur à ce moment. On définit l'<encre> de ce point sur une échelle de 0 à 15.

Le paramètre facultatif <mode d'encre> définit le mode d'interaction entre la couleur utilisée et celle de l'écran. Voici les quatre modes possibles :

0 : Normal
1 : XOR (OU exclusif)
2 : AND (ET)
3 : OR (OU)

Mots clés associés : **GRAPHICS PEN**, **PLOT**

POKE

POKE <adresse>,<nombre entier>

```
10 FOR m=49152 TO 65535
20 POKE m,100
30 NEXT
run
```

COMMANDE : Inscrit la valeur correspondant au <nombre entier> (compris entre 0 et 255) directement dans la case de la mémoire vive (RAM) du Z80 dont l'<adresse> est indiquée.

Commande à utiliser avec précaution !

Mots clés associés : **PEEK**

POS

POS (# <numéro de canal>)

```
10 MODE 1:BORDER 0:LOCATE 8,2
20 PRINT"utilisez les fleches droite/gauche "
30 WINDOW 1,40,12,12:CURSOR 1,1
40 FOR n=1 TO 19:PRINT CHR$(9);:NEXT
50 IF INKEY(1)<>-1 THEN PRINT CHR$(9);
60 IF INKEY(8)<>-1 THEN PRINT CHR$(8);
70 LOCATE #1,2,24
80 PRINT #1,"curseur texte,";
90 PRINT #1,"position horizontale=";
100 PRINT #1,POS(#0):GOTO 50
run
```

FONCTION : Calcule la **POS**ition du curseur de texte sur l'axe horizontal, à partir du bord gauche de la fenêtre. Le <numéro de canal> doit obligatoirement être précisé ; il ne prend pas la valeur #**Ø** par défaut.

POS(#8) calcule la position horizontale du chariot de l'imprimante par rapport à la marge de gauche (de coordonnée 1).

POS(#9) calcule la position logique du canal d'unité de disquettes, c'est-à- dire le nombre de caractères transmis depuis le dernier « retour chariot ».

Mots clés associés : **VPOS**, **WINDOW**

PRINT

PRINT [#<numéro de canal>,][<liste de:<article à imprimer>]

```
10 a$="petite"
20 b$="Ceci est une longue chaine de caracteres"
30 PRINT a$;a$
40 PRINT a$,a$
50 PRINT
60 PRINT b$;b$
70 PRINT b$,b$
run
```

COMMANDE : Transmet la >liste de:<article à imprimer> ou à afficher sur le canal indiqué, (#**Ø** par défaut).

Le point-virgule indique à l'ordinateur qu'un article doit être imprimé immédiatement à la suite du précédent. Toutefois, s'il est trop long pour tenir sur la même ligne, l'ordinateur passe tout de même à la ligne.

La virgule indique qu'un article doit être positionné à la tabulation suivante. Toutefois, si l'impression ou l'affichage de l'article précédent déborde sur la tabulation indiquée, l'ordinateur décale le nouvel article d'une tabulation supplémentaire.

PRINT SPC
PRINT TAB

PRINT [#<numéro de canal>,][<liste de:<article à imprimer>][;]
[SPC (<nombre entier>)][<liste de:<article à imprimer>]

PRINT [#<numéro de canal>,][<liste de:<article à imprimer>][;]
[TAB (<nombre entier>)][<liste de:<article à imprimer>]

```
10 PRINT"ceci est l'instruction SPC"
20 FOR x=6 TO 15
30 PRINT SPC(5)"a";SPC(x);"b"
40 NEXT
50 PRINT"ceci est l'instruction TAB"
60 FOR x=6 TO 15
70 PRINT TAB(5)"a";TAB(x);"b"
80 NEXT
run
```

SPC ménage le nombre d'espaces vides indiqué par le <nombre entier> avant d'imprimer ou d'afficher l'article indiqué, à condition que ce dernier tienne intégralement sur la ligne. Il est donc inutile d'utiliser le point-virgule avec la commande **SPC**.

TAB ménage, à partir de la marge de gauche, le nombre d'espaces vides indiqué avant d'imprimer ou d'afficher l'article désigné, à condition que ce dernier tienne sur la ligne. Le point-virgule est donc inutile après **TAB**. Si le curseur a déjà dépassé la position demandée, un changement de ligne est effectué avant la tabulation.

PRINT USING

PRINT [# <numéro de canal>][<liste de:<article à imprimer>][;]
[**USING**<modèle de format>][<séparateur> <expression>]

```
10 FOR x=1 TO 10
20 n=100000*(RND↑5)
30 PRINT"marchandise";USING "########,##";n
40 NEXT
run
```

PRINT USING permet de définir le format d'impression ou d'affichage d'une expression transmise par la commande **PRINT**. On définit pour cela le <modèle de format> sous lequel on désire voir apparaître l'expression. On utilise comme <séparateur> soit une virgule, soit un point-virgule. Le <modèle de format> est une chaîne de caractères composée des « indicateurs de champ » suivants :

Formats numériques

Dans un nombre :

Chaque signe # indique l'emplacement d'un chiffre.
Exemple : ######

. Indique l'emplacement du point décimal (équivalent à notre virgule).
Exemple : ######.##

, (Réserve un espace). Ce signe, ne pouvant figurer qu'immédiatement avant le point décimal, indique que les chiffres situés à gauche du point décimal seront disposés par groupes de trois (correspondant aux milliers) séparés entre eux par une virgule.
Exemple : ######,.##

Encadrement d'un nombre :

££ (Réserve deux espaces). Indique que le signe £ apparaîtra immédiatement avant le premier chiffre ou le point décimal, c'est-à-dire sur l'un des emplacements réservés aux chiffres.
Exemple : **££**######,.##

** (Réserve deux espaces). Indique que tous les espaces vides situés avant le nombre seront comblés par les astérisques.
Exemple : **######,.##

**£ (Réserve trois espaces). Additionne les options ** et ££, c'est-à-dire les astérisques en tête et le signe £ précédant immédiatement le nombre.
Exemple : **£######,.##.

$$** (Réserve deux espaces). Indique que le signe **$ apparaîtra immédiatement à gauche du premier chiffre ou du point décimal, c'est-à-dire sur l'un des emplacements réservés aux chiffres.
Exemple : $$######,.##

****$** (Réserve trois espaces). Additionne les options ** et **$$**, c'est-à-dire les astérisques en tête et le signe **$** précédant immédiatement le nombre.
Exemple : **$######,.##

\+ Indique qu'on désire voir figurer le signe du nombre. Ce signe apparaîtra avant le nombre si le + est situé au début du <modèle de format> et après le nombre s'il est situé à la fin.
Exemple : +####.####

\- Le signe - ne peut figurer qu'A LA FIN du masque. Il demande la présence du signe - après tout nombre ou tout exposant négatifs. En l'absence de cette spécification, le signe - apparaît par défaut avant le nombre négatif.
Exemple : ####.####-

↑↑↑↑ Indique que le nombre doit apparaître en exposant. Les signes ↑↑↑↑ se placent APRES le dernier emplacement des chiffres, mais AVANT tout signe + ou - final.
Exemple : #.####↑↑↑↑+

La longueur maximale du <modèle de format> d'un nombre est de 20 caractères. Les nombres sont arrondis au nombre de signes indiqué.

Si un format est trop petit pour contenir l'expression saisie :

```
PRINT USING "####";12345678
```

...celle-ci n'est pas tronquée, mais apparaît dans son intégralité, précédée du signe **%** indiquant un « format erroné ».

Format d'une chaîne alphanumérique

```
10 CLS:a$="abcdefghijklmnopqrst"
20 PRINT"chaine alphanum.= ";a$
30 PRINT:PRINT"Avec ! = ";
40 PRINT USING "!";a$
50 PRINT:PRINT "Avec \espaces\ = ";
60 PRINT USING "\        \";a$
70 PRINT:PRINT "Avec & = ";
80 PRINT USING "&";a$
90 GOTO 90
run
```

! Indique que seul le premier caractère de la chaîne doit apparaître.
Exemple : !

\\<espaces> \\
Indique que seuls les x premiers caractères de la chaîne doivent apparaître, x étant égal à la longueur du format (barres comprises).
Exemple : \\   \\

& Indique que la chaîne doit apparaître « telle quelle ».
Exemple : **&**

Le <modèle de format> d'une chaîne ne peut excéder 255 caractères.

Tout <modèle de format> peut être représenté par une variable alphanumérique, comme le montre l'exemple suivant :

```
10 a$="FF######,.##"
20 b$="!"
30 PRINT USING a$;12345.6789;
40 PRINT USING b$;"centimes"
run
```

Pour plus de détails concernant les formats, voir la deuxième partie du chapitre intitulé « A vos heures de loisir... ».

Mots clés associés : **SPC, TAB, USING, ZONE**

RAD

RAD

```
RAD
```

COMMANDE : Etablit le mode de calcul en **RAD**ians. En BASIC, il est adopté par défaut.

Mots clés associés: **ATN**, **COS**, **DEG**, **SIN**, **TAN**

RANDOMIZE

RANDOMIZE [<expression numérique>]

```
RANDOMIZE 123.456
PRINT RND
 0.258852139
```

COMMANDE : Donne une valeur aléatoire calculée à partir de l'<expression numérique> indiquée. Le générateur de nombres aléatoires fournit une séquence pseudo-aléatoire dans laquelle chaque nombre dépend du précédent. La séquence elle-même est prédéterminée. Si la valeur initiale n'est pas précisée dans la commande, c'est l'utilisateur qui l'entrera en cours d'exécution. **RANDOMIZE TIME** fournit une séquence pratiquement imprévisible.

Mots clés associés : **RND**

READ

READ <liste de:<variable>

```
10 FOR n=1 TO 8
20 READ a$,c
30 PRINT a$;" ";:SOUND 1,c:NEXT
40 DATA voici,478,les,426,8,379,notes
50 DATA 358,de,319,la,284,gamme,253,.,239
run
```

COMMANDE : Lit les données contenues dans une instruction **DATA** et les assigne à des variables. **READ** passe automatiquement d'une donnée à la suivante. La commande **RESTORE** permet de revenir à une commande **DATA** antérieure.

Pour plus de détails, voir la deuxième partie du chapitre intitulé « A vos heures de loisir... ».

Mots clés associés : **DATA**, **RESTORE**

RELEASE

RELEASE <canaux sonores>

```
10 SOUND 65,1000,100
20 PRINT"appuyez [R] pour liberer la note"
30 IF INKEY(50)=-1 THEN 30
40 RELEASE 1
run
```

COMMANDE : Libère les canaux sonores bloqués par la commande **SOUND**.

Le paramètre <canaux sonores> prend les valeurs suivantes :

1 : Libère le canal A
2 : Libère le canal B
3 : Libère les canaux A et B
4 : Libère le canal C
5 : Libère les canaux A et C
6 : Libère les canaux B et C
7 : Libère les canaux A, B et C

Pour en savoir plus sur le mode sonore, consultez la deuxième partie du chapitre intitulé : « A vos heures de loisir... ».

Mots clés associés : **SOUND**

REM

REM <texte>

```
10 REM CHASSE AUX ENVAHISSEURS DANS L'HYPERESPACE
INTERGALACTIQUE
20 REM         COPYRIGHT by AMSOFT
```

COMMANDE : Insère une **REM**arque dans le programme. BASIC ne tient pas compte du <texte> situé sur la ligne à droite de **REM**, même si celui-ci comprend un séparateur d'instructions " : " ou tout autre code.

On peut remplacer **:REM** par une apostrophe ' dans tous les cas, SAUF à l'intérieur d'une instruction **DATA**.

Mots clés associés : Aucun

REMAIN

REMAIN (<numéro de chronomètre>)

```
10 AFTER 500,1 GOSUB 40
20 AFTER 100,2 GOSUB 50
30 PRINT"Le programme tourne":GOTO 30
40 REM ce sous programme ne sera plus appele dans la mesure
ou il a ete rendu innoperant en ligne 70
50 PRINT:PRINT "Le chronometre 1 va ";
60 PRINT"etre supprime par REMAIN."
70 PRINT"il restait";REMAIN(1);"unites de temps au chrono 1
run
```

FONCTION : Lit le temps restant à décompter par le chronomètre indiqué (de 0 à 3), avant de le désactiver. Pour plus de détails concernant les interruptions, consultez la deuxième partie du chapitre « A vos heures de loisir... ».

Mots clés associés : **AFTER**, **DI**, **EI**, **EVERY**

RENUM

RENUM [<nouveau numéro de ligne>][,[<ancien numéro de ligne>][,<incrément>]]

```
10 CLS
20 REM cette ligne deviendra:ligne 123
30 REM cette ligne deviendra:ligne 124
40 REM cette ligne deviendra:ligne 125
RENUM 123,20,1
LIST
```

COMMANDE : **RENUM**érote les lignes d'un programme.

L'<ancien numéro de ligne> est un paramètre indiquant la ligne du programme à laquelle on désire commencer la renumérotation. En l'absence de ce paramètre, toutes les lignes du programme seront renumérotées. Le <nouveau numéro de ligne> indique le nouveau numéro de la première ligne renumérotée (**10** par défaut). L'<incrément> indique l'espacement désiré entre deux lignes (**10** par défaut). **RENUM** opère les réajustements nécessaires à l'intérieur des instructions d'appel telles que **GOTO** et **GOSUB**. En revanche, il laisse inchangé les numéros de lignes contenus dans des chaînes de caractères apparaissant dans les commandes **KEY**, **REM**, **CHAIN** et **CHAIN MERGE**. Les numéros de lignes doivent être compris entre **1** et **65535**.

Mots clés associés : **DELETE**, **LIST**

RESTORE

RESTORE [<numéro de ligne>]

```
10 READ a$:PRINT a$;" ";
20 RESTORE 50
30 FOR t=1 TO 500:NEXT:GOTO 10
40 DATA les data recuperes peuvent etre lus encore
50 DATA et encore
run
```

COMMANDE : Ramène le pointeur sur l'instruction **DATA** indiquée. En l'absence de paramètre, le pointeur retourne à la première instruction **DATA** du programme.

Pour plus de détails, consultez la deuxième partie du chapitre « A vos heures de loisir... ».

Mots clés associés : **DATA, READ**

RESUME

RESUME [<numéro de ligne>]

```
10 ON ERROR GOTO 60
20 FOR x=10 TO 0 STEP-1:PRINT 1/x:NEXT
30 END
40 PRINT"je viens ici en cas d'erreur"
50 END
60 PRINT"erreur No.";ERR;"a la ligne";ERL
70 RESUME 40
run
```

COMMANDE : Reprend l'exécution d'un programme après la détection et le traitement d'une erreur par la commande **ON ERROR GOTO**. Si aucun <numéro de ligne> n'est indiqué, l'exécution du programme reprend à la ligne contenant l'erreur détectée. Supprimez ce paramètre dans l'exemple ci-dessus, puis faites tourner le programme.

```
70 RESUME
run
```

Mots clés associés : **DERL, ERL, ERR, ERROR, ON ERROR GOTO, RESUME NEXT**

RESUME NEXT

RESUME NEXT

```
10 ON ERROR GOTO 90
20 PRINT"tapez [ENTER] a chaque fois"
30 INPUT "1";a
40 INPUT "2";a
50 inpot "3";a:REM erreur de syntaxe
60 INPUT "4";a
70 INPUT "5";a
80 END
90 PRINT"erreur No.";ERR;"a la ligne";ERL
100 RESUME NEXT
run
```

COMMANDE : Reprend l'exécution d'un programme après la détection et le traitement d'une erreur par la commande **ON ERROR GOTO**.

L'exécution du programme reprend à partir de la ligne suivant immédiatement la ligne erronée.

Mots clés associés : **DERR**, **ERR**, **ERROR**, **ON ERROR GOTO**, **RESUME**

RETURN

RETURN

```
10 GOSUB 50:PRINT "Apres le GOSUB":END
50 FOR n=1 TO 20
60 PRINT"sous-programme"
70 NEXT:PRINT
80 RETURN
run
```

COMMANDE : Indique la fin d'un sous-programme. Aprés l'exécution d'un sous-programme, BASIC retourne à l'instruction suivant immédiatement l'appel **GOSUB** correspondant.

Mots clés associés : **GOSUB**

RIGHT$

RIGHT$ (<chaîne alphanumérique>,<longueur requise>)

```
10 MODE 1:a$="ordinateur CPC 6128"
20 FOR n=1 TO 16:LOCATE 41-n,n
30 PRINT RIGHT$(a$,n)
40 NEXT
run
```

FONCTION : Extrait un certain nombre de caractères (entre 0 et 255) à gauche d'une <chaîne alphanumérique>. Si la chaîne est plus courte que la <longueur requise>, elle est utilisée entièrement.

Mots clés associés : **LEFT$**, **MID$**

RND

RND [(<expression numérique>)]

```
10 RANDOMIZE
20 FOR x=1 TO -1 STEP -1
30 PRINT"parametre RND=";x
40 FOR n=1 TO 6
50 PRINT RND(x)
60 NEXT n,x
run
```

FONCTION : Fournit le prochain nombre de la séquence pseudo aléatoire en cours lorque l'<expression numérique> est positive ou lorsqu'elle ne figure pas dans la commande.

Lorsque l'<expression numérique> est nulle, **RND** renvoie le dernier nombre généré.

Une valeur négative de l'<expression numérique> lance une nouvelle séquence aléatoire, dont **RND** fournit le premier élément.

Mots clés associés : **RANDOMIZE**

ROUND

ROUND (<expression numérique>[,<nombre de décimales>])

```
10 FOR n=4 TO -4 STEP -1
20 PRINT ROUND(1234.5678,n),
30 PRINT "arrondi a";n;"decimales"
40 NEXT
run
```

FONCTION : Arrondit l'<expression numérique> au nombre de chiffres après la virgule ou de puissances de dix indiqué par le paramètre <nombre de décimales>. Si ce paramètre est négatif, l'expression est arrondie à un entier absolu, suivi d'un nombre de zéros égal à sa valeur absolue.

Mots clés associés : **ABS**, **CINT**, **FIX**, **INT**

RUN

RUN <chaîne alphanumérique>

```
RUN "disc"
```

COMMANDE : Charge et exécute un programme BASIC ou un programme-objet situé sur la disquette. Tout programme déjà présent en mémoire est automatiquement écrasé.

Cette commande permet d'accéder directement aux programmes BASIC protégés.

Mots clés associés : **LOAD**

RUN

RUN [<numéro de ligne>]

```
RUN 200
```

COMMANDE : Exécute le programme BASIC présent en mémoire, en commençant au <numéro de ligne> indiqué ou, à défaut, au début du programme. RUN réinitialise toutes les variables.

Cette commande peut ne pas donner accès aux programmes protégés chargés en mémoire.

Mots clés associés : **CONT**, **END**, **STOP**

SAVE

SAVE <nomfich>[,<type de fichier>][,<paramètres binaires>]

```
SAVE "fichdisc.xyz"
```

...sauvegarde le fichier en mode BASIC non protégé.

```
SAVE "fichdisc.xyz",P
```

...sauvegarde le fichier en mode BASIC Protégé.

```
SAVE "fichdisc.xyz",A
```

...sauvegarde le fichier en mode ASCII.

```
SAVE "fichdisc.xyz",B,8000,3000,8001
```

...sauvegarde le fichier en mode Binaire. Dans notre exemple, le programme sera stocké en mémoire à partir de l'adresse **8000** et occupera **3000** octets. L'adresse (facultative) du point d'entrée est **8001**.

COMMANDE : Sauvegarde sur disquette le programme se trouvant actuellement en mémoire. Une zone mémoire chargée sur disquette s'appelle un fichier Binaire. Voici quels sont les différents paramètres Binaires :

<adresse du début>,<taille du fichier>[,<point d'entrée>]

Il est possible de sauvegarder la mémoire d'écran sous forme de fichier Binaire. Cette opération, appelée « vidage d'écran », s'effectue au moyen de la commande suivante :

```
SAVE "ecran",B,&C000,&4000
```

On obtient ensuite son rechargement à l'écran en entrant la commande :

```
LOAD "ecran"
```

Mots clés associés : **CHAIN**, **CHAIN MERGE**, **LOAD**, **MERGE**, **RUN**

SGN

SGN (<expression numérique>)

```
10 FOR n=200 TO -200 STEP -20
20 PRINT" SGN renvoi";
30 PRINT SGN(n);"pour une valeur de:";n
40 NEXT
run
```

FONCTION : Etablit le **SiGN**e de l'<expression numérique>. SGN renvoie les valeurs : -1 (si l'expression est négative), 0 (si elle est nulle) et 1 (si elle est positive).

Mots clés associés : **ABS**

SIN

SIN (<expression numérique>)

```
10 CLS:DEG:ORIGIN 0,200
20 FOR n=0 TO 720
30 y=SIN(n)
40 PLOT n*640/720,198*y:NEXT
50 GOTO 50
run
```

FONCTION : Calcule le **SIN**us de l'<expression numérique> indiquée.

On peut exprimer l'argument en degrés ou en radians en utilisant, respectivement, les fonctions **DEG** et **RAD**.

Mots clés associés : **ATN**, **COS**, **DEG**, **RAD**, **TAN**

SOUND

SOUND <état de canal>,<période sonore>[,<durée>[,<volume>[,<enveloppe de volume>[,<enveloppe de tonalité>[,<période du bruit>]]]]]

```
10 FOR z=0 TO 4095
20 SOUND 1,z,1,12
30 NEXT
run
```

COMMANDE : Permet la programmation d'un son, à l'aide des paramètres suivants :

Paramètre 1 : <état de canal>

L'<état de canal> admet pour valeur des entiers compris entre 1 et 255. La conversion en binaire de ce paramètre donne la signification de chaque bit, selon la table de correspondance suivante :

Bit 0 (1 en décimale) : sortir le son sur le canal A (Bit de poids faible)
Bit 1 (2 en décimale) : sortir le son sur le canal B
Bit 2 (4 en décimale) : sortir le son sur le canal C
Bit 3 (8 en décimale) : rendez-vous avec le canal A
Bit 4 (16 en décimale) : rendez-vous avec le canal B
Bit 5 (32 en décimale) : rendez-vous avec le canal C
Bit 6 (64 en décimale) : bloquer un canal sonore
Bit 7 (128 en décimale) : vider un canal sonore (Bit de poids fort)

L'<état de canal> **68**, par exemple, aura l'effet suivant :

Sortie sur le canal C (4), à l'état bloqué (64).

Paramètre 2 : <période sonore>

Ce paramètre établit la hauteur du son, c'est-à-dire la « note » produite (par exemple Do, Ré, Mi, Fa, Sol). Chaque note se définit par une valeur numérique représentant sa <période sonore> (Voir le chapitre « Pour information... ».

Paramètre 3 : <durée>

Ce paramètre établit la longueur, ou « durée », du son. 1 unité correspond à un centième de seconde. La <durée> du son prend par défaut la valeur 20, c'est-à-dire un cinquième de seconde.

En cas de paramètre de <durée> nul, la longueur du son sera celle de l'enveloppe de volume indiquée.

Si le paramètre de <durée> est négatif, l'enveloppe de volume sera répétée pendant un nombre de fois égal à la valeur absolue du paramètre.

Paramètre 4 : <volume>

Ce paramètre établit le volume sonore initial d'une note. Il peut prendre une valeur entre 0 (volume nul) et 15 (volume maximal). L'ordinateur choisit par défaut la valeur 12.

Paramètre 5 : <enveloppe de volume>

Il est possible de moduler le volume d'une note durant son exécution à l'aide de la commande **ENV**. Cette commande vous permet de définir à l'avance un maximum de quinze enveloppes différentes, codées de 1 à 15. Le paramètre <enveloppe de volume> de la commande **SOUND** permet ensuite de sélectionner une de ces enveloppes prédéfinies.

Voir la commande **ENV**

Paramètre 6 : <enveloppe de tonalité>

Des variations de période, donc de hauteur, peuvent être obtenues durant l'exécution d'une note par l'intermédiaire de la commande **ENT**. Cette commande vous permet de définir à l'avance un maximum de 15 enveloppes de tonalité différentes, codées de 1 à 15. Le paramètre <enveloppe de tonalité> de la commande **SOUND** permet ensuite de sélectionner une de ces enveloppes prédéfinies. Si, dans la commande ENT, vous avez utilisé un numéro d'enveloppe négatif, la valeur absolue de ce nombre devra alors être prise comme paramètre de la commande **SOUND**.

Voir la commande **ENT**.

Paramètre 7 : <période de bruit>

Vous disposez d'un choix de bruits blancs pouvant être ajoutés ou supprimés du signal sonore à l'aide du paramètre <période du bruit> (valeur comprise entre 0 et 31).

Pour en savoir plus sur le mode sonore, lire la deuxième partie du chapitre intitulé « A vos heures de loisir... »

Mots clés associés : **ENT**, **ENV**, **ON SQ GOSUB**, **RELEASE**, **SQ**

SPACE$

SPACE$ (<nombre entier>)

```
10 MODE 1
20 PRINT"met 9 espaces entre vous";
30 PRINT SPACE$(9);
40 PRINT"et moi!"
run
```

FONCTION : Crée une chaîne d'espaces de la longueur indiquée (de 0 à 255)

Mots clés associés : **SPC**, **STRING$**, **TAB**

SPC

(Voir **PRINT SPC**)

SPEED INK

SPEED INK <période 1>,<période 2>

```
10 BORDER 7,18
20 FOR i=30 TO 1 STEP -1
30 SPEED INK i,i
40 FOR t=1 TO 700:NEXT t,i
run
```

COMMANDE : Permet d'établir la période d'alternance lorsqu'une instruction **INK** ou **BORDER** prescrit l'utilisation de deux couleurs intermittentes. Les durées respectives d'utilisation de la première et de la seconde couleur sont indiquées en cinquantièmes de secondes par les paramètres <période 1> et <période 2>.

Lors du choix des paramètres, pensez aux risques d'effets secondaires hypnotiques !

Mots clés associés : **BORDER**, **INK**

SPEED KEY

SPEED KEY <délai initial>,<intervalle inter-répétitions>

```
10 CLS:FOR K=7 TO 1 STEP -2
20 PRINT"Entrez votre nom ,puis[RETURN]"
30 SPEED KEY k,k
40 LINE INPUT a$:NEXT
50 PRINT"quel drole de nom !"
run
```

COMMANDE : Etablit la vitesse de répétition automatique du clavier. Le paramètre <délai initial> fixe le temps de réaction (mesuré en cinquantièmes de secondes) entre l'enfoncement de la touche et le début de la répétition automatique. L'<intervalle inter-répétitions> établit le laps de temps séparant les répétitions.

La commande **SPEED KEY** ne concerne que les touches pour lesquelles la répétition automatique existe implicitement ou celles pour lesquelles cette fonction a été programmée au moyen de la commande **KEY DEF**.

Avant de définir une répétition automatique à très faible <délai initial>, il est prudent de programmer une des touches numériques afin de pouvoir rétablir les paramètres par défaut de la fonction **SPEED KEY** (30,2). Voici comment procéder :

```
KEY 0,"SPEED KEY 30,2"+CHR$(13)
```

Il suffira, pour revenir à l'état initial, d'actionner la touche **Ø** du pavé numérique.

Mots clés associés : **KEY DEF**

SPEED WRITE

SPEED WRITE <nombre entier>

```
SPEED WRITE 1
```

COMMANDE : Indique la vitesse de transmission des données de l'ordinateur vers un lecteur de cassettes, le cas échéant. Cette vitesse est soit de 2000 bauds (bits par seconde) si le paramètre est égal à 1, soit, par défaut, de 1000 bauds si celui-ci est égal à 0. Lors du chargement d'un fichier enregistré sur cassette, l'ordinateur choisit automatiquement la bonne vitesse de lecture.

SPEED WRITE Ø est le débit assurant la meilleure fiabilité de transfert.

La commande **SPEED WRITE** ne s'applique pas aux unités de disquettes.

Mots clés associés : **OPENOUT**, **SAVE**

SQ

SQ (<numéro de canal>)

```
10 SOUND 65,100,100
20 PRINT SQ(1)
run
 67
```

FONCTION : Indique l'état de la file d'attente (Sound Queue) dans un canal sonore donné. Le <numéro de canal> doit être une expression numérique entière prenant les valeurs :

1 : canal A
2 : canal B
4 : canal C

Cette fonction fournit un entier correspondant aux bits de signification suivants :

Bits 0, 1 et 2 : nombres d'entrées libres dans la file
Bits 3, 4 et 5 : état du rendez-vous au début de la file
Bit 6 : la tête de la file est bloquée
Bit 7 : le canal est en activité

... où le Bit 0 représente le bit de poids faible, et le Bit 7 celui de poids fort.

On peut constater que si le Bit 6 est à 1, le Bit 7 ne peut pas l'être. De même si les Bits 3, 4 ou 5 sont à 1, les Bits 6 et 7 ne peuvent pas l'être.

Pour plus de détails, voir la deuxième partie du chapitre intitulé « A vos heures de loisir... »

Mots clés associés : **ON SQ GOSUB**, **SOUND**

SQR

SQR (<expression numérique>)

```
PRINT SQR(9)
 3
```

FONCTION : Fournit la racine carrée (**SQ**uare **R**oot) de l'expression numérique indiquée.

Mots clés associés : Aucun

STEP

(voir **FOR**)

STOP

STOP

```
10 FOR n=1 TO 30:PRINT n:NEXT
20 STOP
30 FOR n=31 TO 60:PRINT n:NEXT
run
cont
```

COMMANDE : Interrompt un programme, tout en laissant à l'utilisateur la possibilité d'en reprendre l'exécution au moyen de la commande **CONT**. **STOP** ; permet ainsi d'interrompre un programme à un endroit donné afin d'effectuer une mise au point.

Mots clés associés : **CONT**, **END**

STR$

STR$ (<expression numérique>)

```
10 a=&FF:REM 255 hexadecimal
20 b=&X1111:REM 15 binaire
30 c$="***"
40 PRINT c$+STR$(a+b)+c$
run
*** 270***
```

FONCTION : Fournit sous forme de chaîne alphanumérique la représentation décimale de l'<expression numérique> indiquée.:

Mots clés associés : **BIN$**, **DEC$**, **HEX$**, **VAL**

STRING$

STRING$ (<longueur>,<caractère>)

```
PRINT STRING$(40,"*")
****************************************
```

FONCTION : Fournit une chaîne de caractères de la longueur indiquée (entre 0 et 255) constituée par la répétition d'un même caractère. L'exemple ci-dessus peut également s'écrire :

```
PRINT STRING$(40,42)
****************************************
```

...où le <caractère> **42** correspond à la valeur en code ASCII du caractère « * ». Cette notation équivaut donc à **PRINT STRING$(40,CHR$(42))**.

Mots clés associés : **SPACE$**

SWAP

(voir **WINDOW SWAP**)

SYMBOL

SYMBOL <numéro de caractère>,<liste de:<ligne>

```
10 MODE 1:SYMBOL AFTER 105
20 rangee1=255:REM 11111111 en binaire
30 rangee2=129:REM 10000001 en binaire
40 rangee3=189:REM 10111101 en binaire
50 rangee4=153:REM 10011001 en binaire
60 rangee5=153:REM 10011001 en binaire
70 rangee6=189:REM 10111101 en binaire
80 rangee7=129:REM 10000001 en binaire
90 rangee8=255:REM 11111111 en binaire
100 PRINT"La ligne 110 redefinie la lettre i(105). Tapez que
lques 'i' et regardez !"
110 SYMBOL 105,rangee1,rangee2,rangee3,rangee4,rangee5,range
e6,rangee7,rangee8
run
```

COMMANDE : Redéfinit la forme d'un caractère affiché à l'écran. Chacun des paramètres prend une valeur entière située entre 0 et 255.

Afin d'être en mesure d'attribuer une place en mémoire à un caractère redéfini, l'ordinateur doit avoir été préparé par la commande :

```
SYMBOL AFTER x
```

...où **x** est inférieur ou égal au numéro du caractère à redéfinir.

On entre ensuite la commande SYMBOL, suivie immédiatement du numéro de caractère **x**.

On peut toujours faire apparaître à l'écran le caractère correspondant à la valeur de **x**, même s'il n'est pas accessible par l'intermédiaire du clavier. On utilise pour cela la commande :

```
PRINT CHR$(x)
```

Après **SYMBOL x** viennent les huit paramètres définissant une à une les huit lignes constituant le caractère, en commençant par le haut. Chacun des paramètres prend une valeur comprise entre 0 et 255. C'est la représentation binaire du paramètre qui définit le motif de la ligne correspondante dans le nouveau caractère.

Si, par exemple, on donne au premier paramètre la valeur 1, la représentation en binaire de la première rangée sera alors : 00000001. Le point correspondant au 1 apparaîtra dans le caractère et sera de la couleur définie dans la commande **PEN**. En revanche, les points correspondant aux 0 seront affichés avec la couleur **PAPER**, c'est-à-dire qu'ils n'apparaîtront pas. On peut donc constater que le caractère redéfini dans notre exemple comportera un point dans le coin supérieur droit. Si l'on attribue aux sept autres paramètres les valeurs : **3**, **7**, **15**, **31**,**63**, **Ø**, **Ø**, on obtient la représentation en binaire suivante :

paramètre (rangée) 1 : 00000001 (1 en décimale)
paramètre (rangée) 2 : 00000011 (3 en décimale)
paramètre (rangée) 3 : 00000111 (7 en décimale)
paramètre (rangée) 4 : 00001111 (15 en décimale)
paramètre (rangée) 5 : 00011111 (31 en décimale)
paramètre (rangée) 6 : 00111111 (63 en décimale)
paramètre (rangée) 7 : 00000000 (0 en décimale)
paramètre (rangée) 8 : 00000000 (0 en décimale)

Cette disposition des paramètres transcrits en binaire permet de visualiser la forme du nouveau caractère. On peut affecter ces paramètres au caractère numéro 255, par exemple, au moyen de la commande :

```
SYMBOL 255,1,3,7,15,31,63,0,0
```

On remarque qu'il est inutile de faire figurer les deux derniers paramètres dans la mesure où il sont nuls.

```
SYMBOL 255,1,3,7,15,31,63
```

On peut éviter de convertir en notation décimale les symboles binaires constituant le « dessin » du caractère. Il suffit pour cela d'introduire les paramètres directement en binaire, sans oublier le préfixe **&X**. Dans notre cas :

```
SYMBOL 255,&X00000001,&X00000011,&X00000111,&X00001111
&X00011111,&X00111111
```

Voici maintenant la commande permettant de faire apparaître le caractère :

```
PRINT CHR$(255)
```

Si les paramètres ci-dessus sont affectés à un des caractères du clavier, le caractère redéfini apparaîtra chaque fois qu'on actionnera la touche correspondante, de même qu'à chaque commande **PRINT** le concernant. Ce nouveau caractère sera accepté par le BASIC comme l'équivalent du caractère remplacé.

Pour plus d'informations concernant la redéfinition de caractères, voir la deuxième partie du chapitre intitulé « A vos heures de loisir... ».

Mots clés associés : **HIMEM**, **MEMORY**, **SYMBOL AFTER**

SYMBOL AFTER

SYMBOL AFTER <nombre entier>

```
10 CLS
20 SYMBOL AFTER 115
30 PRINT"La ligne 40 redefinie la lettre s ";
40 SYMBOL 115,0,56,64,64,48,8,8,112
50 PRINT"en s"
60 PRINT"on revient a l'etat normal en tapant:"
70 PRINT"SYMBOL AFTER 240"
run
```

COMMANDE : Fixe la limite inférieure des numéros de caractères redéfinissables (de 0 à 255). La valeur par défaut du nombre entier est de 240, auquel cas on dispose de 16 caractères redéfinissables (entre 240 et 255). Lorsque le <nombre entier> a pour valeur **32**, tous les caractères situés entre 32 et 255 sont redéfinissables. La commande **SYMBOL AFTER 256** interdit donc toute redéfinition de caractère.

La commande **SYMBOL AFTER** rétablit la valeur par défaut de tous les caractères précédemment redéfinis.

La commande **SYMBOL AFTER** ne peut PAS fonctionner si la valeur de **HIMEM** a été précédemment modifiée par une commande MEMORY ou par l'ouverture d'un fichier au moyen de **OPENIN** ou **OPENOUT**. L'ordinateur affiche dans ce cas le message d'erreur « Improper argument », (à moins que l'état précédent ne soit **SYMBOL AFTER 256**).

Pour plus de détails concernant les caractères redéfinissables, consultez la deuxième partie du chapitre intitulé « A vos heures de loisir... »

Mots clés associés : **HIMEM**, **MEMORY**, **SYMBOL**

TAB

(voir **PRINT TAB**)

TAG

TAG [# <numéro de canal>]

```
10 INPUT "entrez votre nom";a$:CLS
20 PRINT"Quel va et vient ";a$;" !!"
30 TAG
40 x=LEN(a$)*17:y=50+RND*300:MOVE -x,y
50 FOR f=-x TO 640 STEP RND*7+3
60 MOVE f,y:PRINT" ";a$;:FRAME:NEXT
70 FOR b=640 TO -x STEP -RND*7+3
80 MOVE b,y:PRINT a$;" ";:FRAME:NEXT
90 GOTO 40
run
```

COMMANDE : Ecrit le texte spécifié à la position du curseur graphique. Cette commande permet d'introduire du texte et des symboles dans un graphique et de les déplacer pixel par pixel plutôt que caractère par caractère. Le numéro de canal prend par défaut la valeur # **Ø**.

L'extrémité gauche de la chaîne de caractères se positionne sur le curseur graphique (**T**ext **A**t **G**raphics). Les caractères de contrôle non visualisés tels que le changement de ligne ou le retour chariot n'auront aucun effet à l'écran si l'instruction **PRINT** est terminée par un point-virgule ; dans le cas contraire, ils apparaîtront sous leur forme graphique.

Si l'indicateur de canal est # **Ø** (par défaut), BASIC annule la commande **TAG** lors du retour en mode direct.

Mots clés associés : **TAGOFF**

TAGOFF

TAGOFF [# <numéro de canal>]

```
10 MODE 2:TAG:REM texte aux coordonnees graphiques
20 annee=1984:FOR x=1 TO 640 STEP 60
30 MOVE x,400:DRAWR 0,-350
40 annee=annee+1:PRINT annee;:NEXT
50 TAGOFF:REM retour aux coordonnees texte
60 LOCATE 28,25:PRINT"chiffres annuels"
70 GOTO 70
run
```

COMMANDE : Annule la commande **TAG** concernant le canal indiqué (# **Ø** par défaut). Le texte se trouve donc à nouveau dirigé sur la position du curseur de texte.

Mots clés associés : **TAG**

TAN

TAN (<expression numérique>)

```
PRINT TAN(45)
 1.61977519
```

FONCTION : Calcule la **TAN**gente de l'<expression numérique>, qui doit être comprise entre -200000 et +200000.

On peut exprimer l'argument en degrés ou en radians par l'intermédiaire des fonctions **DEG** et **RAD**, respectivement.

Mots clés associés : **ATN**, **COS**, **DEG**, **RAD**, **SIN**

TEST

TEST (<coordonnée x>,<coordonnée y>)

```
10 CLS
20 PRINT"Vous utilisez le stylo (pen) No:";
30 PRINT TEST(12,394)
40 PRINT"Changez de mode et de stylo";
50 PRINT"... puis faites RUN."
run
```

FONCTION : Place le curseur graphique à la position définie par x et y (en coordonnées absolues) et indique la valeur du paramètre <encre> à cet endroit.

Mots clés associés : **MOVE**, **MOVER**, **TESTR**, **XPOS**, **YPOS**

TESTR

TESTR (<décalage x>,<décalage y>)

```
10 MODE 0:FOR x=1 TO 15:LOCATE 1,x
20 PEN x:PRINT STRING$(10,143);:NEXT
30 MOVE 200,400:PEN 1
40 FOR n=1 TO 23:LOCATE 12,n
50 PRINT"pen";TESTR(0,-16):NEXT
run
```

FONCTION : Place le curseur sur une position de coordonnées x et y par rapport à sa position actuelle et indique la valeur du paramètre <encre> à cet endroit.

Mots clés associés : **MOVE**, **MOVER**, **TEST**, **XPOS**, **YPOS**

THEN

(voir **IF**)

TIME

TIME

```
10 CLS:REM horloge
20 INPUT "heure";heure
30 INPUT "minute";minute
40 INPUT "seconde";seconde
50 CLS:donnee=INT(TIME/300)
60 WHILE heure<13
70 WHILE minute<60
80 WHILE tic<60
90 tic=(INT(TIME/300)-donnee)+seconde
100 LOCATE 1,1
110 PRINT USING "## ";heure;minute;tic
120 WEND
130 tic=0:seconde=0:minute=minute+1
140 GOTO 50
150 WEND
160 minute=0:heure=heure+1
170 WEND
180 heure=1
190 GOTO 60
run
```

FONCTION : Indique le temps écoulé depuis la mise sous tension de l'ordinateur ou la dernière commande **RESET** (les temps de transfert entre l'ordinateur et l'unité de disquette ne sont pas comptés).

A chaque seconde correspond une fois la valeur : **TIME/300**.

Mots clés associés : **AFTER**, **EVERY**, **WEND**, **WHILE**

TO

(Voir **FOR**)

TROFF
TRON

TROFF
TRON

```
10 TROFF:PRINT:PRINT"TROFF"
20 FOR n=1 TO 8
30 PRINT"Le programme tourne":NEXT
40 IF f=1 THEN END
50 TRON:PRINT:PRINT "TRON"
60 f=1:GOTO 20
run
```

COMMANDE : Permet de suivre l'exécution d'un programme par l'affichage de chaque numéro de ligne exécutée. Ce numéro est affiché entre crochets **[]**. Cette fonction s'obtient au moyen de la commande **TRON**. La commande **TROFF** rétablit le mode normal d'exécution. La commande **TRON** est particulièrement précieuse lorsque l'on désire suivre ligne par ligne le déroulement d'un programme afin de corriger une erreur.

Mots clés associés : Aucun

UNT

UNT (<adresse>)

```
PRINT UNT(&FF66)
-154
```

COMMANDE : Convertit l'argument en un nombre entier signé (en représentation : complément à 2) compris entre -32768 et 32767.

Mots clés associés : **CINT**, **FIX**, **INT**, **ROUND**

UPPER$

UPPER$ (<chaîne alphanumérique>)

```
10 CLS:a$="mes petites, comme vous avez grandies !"
20 PRINT UPPER$(a$)
run
```

FONCTION : Recopie la <chaîne alphanumérique> indiquée en remplaçant par des majuscules les caractères alphabétiques (de A à Z) apparaissant en minuscules. Cette fonction s'utilise en particulier pour le traitement d'entrées dans lesquelles se trouvent mélangées des majuscules et des minuscules.

Mots clés associés : **LOWER$**

USING

(voir **PRINT USING**)

VAL

VAL (<chaîne de caractères>)

```
10 CLS:PRINT "Je connais mes tables !"
20 PRINT:PRINT"pressez une touche (1-9)"
30 a$=INKEY$:IF a$="" THEN 30
40 n=VAL(a$):IF n<1 OR n>9 THEN 30
50 FOR x=1 TO 12
60 PRINT n;"X";x;"=";n*x
70 NEXT:GOTO 20
run
```

FONCTION : Fournit la **VAL**eur numérique du ou des premiers caractères (y compris le signe négatif et le point décimal) de la <chaîne alphanumérique> indiquée.

On obtient la valeur 0 lorsque le premier caractère de la chaîne n'est pas un chiffre. Si le signe « - » apparaît en premier caractère ou si celui-ci est un point décimal suivi d'un caractère non numérique, le message d'erreur « **Type mismatch** » (erreur de frappe) (13) s'affiche à l'écran.

Mots clés associés : **STR$**

VPOS

VPOS (# <numéro de canal>)

```
10 MODE 1:BORDER 0:LOCATE 8,2
20 PRINT"utilisez les touches flechees (haut/bas)"
30 WINDOW 39,39,1,25:CURSOR 1,1
40 LOCATE 1,13
50 IF INKEY(0)<>-1 THEN PRINT CHR$(11);
60 IF INKEY(2)<>-1 THEN PRINT CHR$(10);
70 LOCATE #1,3,24
80 PRINT#1,"curseur texte ";
90 PRINT#1,"position verticale =";
100 PRINT#1,VPOS(#0):GOTO 50
run
```

FONCTION : Indique, sur l'axe **V**ertical, la **POS**ition du curseur de texte, à partir du bord supérieur de la fenêtre de texte. L'indicateur de canal doit obligatoirement figurer ; il ne prend pas la valeur # 0 par défaut.

Mots clés associés : **POS**, **WINDOW**

WAIT

WAIT <numéro du port>,<masque>[,<inversion>]

```
WAIT &FF34,20,25
```

COMMANDE : Provoque une attente jusqu'à ce que le port d'entrées-sorties désigné transmette une valeur comprise entre 0 et 255 ; de telle sorte qu'après avoir opéré un **XOR** (OU exclusif) avec le <masque>, puis un **AND** (ET) avec le paramètre d'<inversion>, on obtienne un résultat non nul.

Le BASIC attend jusqu'à ce que la condition soit vérifiée.

Cette commande est à utiliser avec précaution.

Mots clés associés : **INP**, **OUT**

WEND

WEND

```
WEND
```

COMMANDE : Indique la fin d'une section de programme exécutée à l'intérieur d'une boucle **WHILE**. Le BASIC reconnaît automatiquement la commande **WHILE** à laquelle **WEND** est associée.

Mots clés associés : **TIME**, **WHILE**

WHILE

WHILE <expression logique>

```
10 CLS :PRINT "chronometre de 10 secondes":t=TIME
20 WHILE TIME<t+3000
30 SOUND 1,0,100,15
40 WEND:SOUND 129,40,30,15
run
```

COMMANDE : Répète une section de programme tant qu'une condition donnée est vérifiée. Le mot clé **WHILE** indique le début de la section à exécuter tandis que l'<expression logique> définit la condition à vérifier.

Mots clés associés : **TIME**, **WEND**

WIDTH

WIDTH <nombre entier>

```
WIDTH 40
```

COMMANDE : Indique le nombre de caractères par ligne lors d'une sortie sur imprimante. Le BASIC se charge donc d'envoyer automatiquement les retours chariot et les sauts de ligne nécessaires durant l'impression.

En l'absence d'une commande **WIDTH**, l'ordinateur adopte par défaut la valeur **132**. La commande **WIDTH 255** supprime tous les retours chariot et sauts de ligne supplémentaires laissant à l'imprimante le soin de générer elle-même ces caractères. Cependant, les retours chariot et sauts de ligne normaux dus à l'instruction **PRINT** continuent à être envoyés à moins que la commande **PRINT** ne soit terminée par un « ; » ou une virgule.

Mots clés associés : **POS**

WINDOW

WINDOW [# <numéro de canal>,]<gauche>,<droite>,<haut>,<bas>

```
10 MODE 0:BORDER 0:REM cible tv
20 INK 0,0:INK 1,25:INK 2,23:INK 3,21
30 INK 4,17:INK 5,6:INK 6,2:INK 7,26
40 PAPER 0:CLS
50 PAPER 1:WINDOW 2,4,1,18:CLS
60 PAPER 2:WINDOW 5,7,1,18:CLS
70 PAPER 3:WINDOW 8,10,1,18:CLS
80 PAPER 4:WINDOW 11,13,1,18:CLS
90 PAPER 5:WINDOW 14,16,1,18:CLS
100 PAPER 6:WINDOW 17,19,1,18:CLS
110 PAPER 7:WINDOW 2,19,19,25:CLS
120 GOTO 120
run
```

COMMANDE : En mode texte, indique les dimensions d'un canal d'affichage à l'écran (on parle dans ce cas de fenêtre). On veillera à ce que les valeurs des paramètres <gauche>, <droite>, <haut> et <bas> correspondent bien aux coordonnées en vigueur dans le MODE-écran utilisé.

Le <numéro de canal> prendra par défaut la valeur # **0**.

Pour plus de détails concernant les fenêtres, voir la deuxième partie du chapitre intitulé « A vos heures de loisir... ».

Mots clés associés : **WINDOW SWAP**

WINDOW SWAP

WINDOW SWAP <numéro de canal>,<numéro de canal>

```
10 MODE 1:INK 1,24:INK 2,9:INK 3,6
20 WINDOW 21,40,13,25:PAPER 3
30 WINDOW #1,1,20,1,12:PAPER #1,2
40 CLS:PRINT #1," Fenetre No"
50 CLS #1:PRINT #1," Fenetre No 1"
60 LOCATE 1,6
70 PRINT" Fenetre Rouge (0)";SPC(2)
80 LOCATE #1,1,6
90 PRINT #1," Fenetre Verte (1)"
100 FOR t=1 TO 1000:NEXT
110 WINDOW SWAP 0,1:GOTO 60
run
```

COMMANDE : Intervertit la première fenêtre et la seconde.

Les deux <numéros de canal> doivent obligatoirement figurer sans être précédés, dans ce cas précis, de l'indicateur de canal #.

Cette commande permet de diriger les messages BASIC sur un autre canal que celui par défaut # 0.

Pour plus de détails concernant les fenêtres, voir la deuxième partie du chapitre intitulé « A vos heures de loisir... ».

Mots clés associés : **WINDOW**

WRITE

WRITE [# <numéro de canal>,][<données à écrire>]

```
10 REM ecrit des donnees sur la disquette
20 INPUT "donnez-moi un nombre";a
30 INPUT "donnez-moi une chaine de caracteres";a$
40 OPENOUT "NOMFICH"
50 WRITE #9,a,a$
60 CLOSEOUT:PRINT "Les donnees sont sauvees sur disquette"
run
```

COMMANDE : Affiche ou écrit (**WRITE**) des données sur le canal indiqué. Deux articles distincts doivent être séparés par une virgule et les chaînes de caractères sont placées entre guillemets.

Dans notre exemple, les données entrées seront écrites sur le canal # 9, c'est-à-dire enregistrées sur disquette.

Pour retrouver les données, on utilisera le programme ci-dessous :

```
10 REM retrouve les donnees sur la disquette
20 OPENIN "NOMFICH":INPUT #9,a,a$
30 CLOSEIN:PRINT"les 2 donnees sont:"
40 PRINT:PRINT a,a$
run
```

Mots clés associés : **INPUT**, **LINE INPUT**

XOR

<argument> XOR <argument>

```
IF "alain" < "bernard" XOR "chien" > "chat" THEN PRINT
 "vrai" ELSE PRINT "faux"
faux
IF "bernard" < "alain" XOR "chat" > "chien" THEN PRINT
 "vrai" ELSE PRINT "faux"
faux
IF "alain" < "bernard" XOR "chat" > "chien" THEN PRINT
 "vrai" ELSE PRINT "faux"
vrai
....
PRINT 1 AND 1
0
PRINT 0 AND 0
0
PRINT 1 AND 0
1
```

OPERATEUR : Effectue bit à bit l'opération booléenne XOR (OU exclusif) sur des entiers. Lorsque les bits des deux arguments ne sont pas identiques, le bit résultant vaut 1.

Pour plus de détails concernant les opérateurs logiques, voir la deuxième partie du chapitre intitulé « A vos heures de loisir... ».

Mots clés associés : **AND**, **OR**, **NOT**

XPOS

XPOS

```
10 MODE 1:DRAW 320,200
20 PRINT"POSition X du curseur graphique=";
30 PRINT XPOS
run
```

FONCTION : Indique, sur l'axe horizontal (**X**), la **POS**ition du curseur graphique.

Mots clés associés : **MOVE**, **MOVER**, **ORIGIN**, **YPOS**

YPOS

YPOS

```
10 MODE 1:DRAW 320,200
20 PRINT"POSition Y du curseur graphique=";
30 PRINT YPOS
run
```

FONCTION : Indique, sur l'axe vertical (**Y**), la **POS**ition du curseur graphique.

Mots clés associés : **MOVE**, **MOVER**, **ORIGIN**, **XPOS**

ZONE

ZONE <nombre entier>

```
10 CLS:FOR z=2 TO 20
20 ZONE z
30 PRINT "X","X ZONE =";z:NEXT
run
```

COMMANDE : Modifie la largeur de la tabulation désignée par la virgule dans la commande **PRINT**. La largeur des zones d'affichage ou d'impression, (de 13 caractères par défaut), peut ainsi prendre une valeur entière quelconque entre 1 et 255.

Mots clés associés : **PRINT**

Chapitre 4
Utilisation des disquettes et des cassettes

Partie 1 : Disquettes

Préparation des disquettes de travail

Ce chapitre traite de la façon de gérer vos disquettes pour un usage quotidien et présente quelques-uns des utilitaires CP/M.

Sujets abordés :

* Duplication des disquettes système
* Introduction au CP/M Plus
* Utilisation des fichiers Help
* Fonctionnement du système avec une ou plusieurs unités
* Copie de fichiers avec PIP
* Utilisation d'une disquette pour un travail en BASIC uniquement
* Progiciel en BASIC
* Installation d'un progiciel CP/M PLus
* Introduction aux extensions graphiques GSX
* Exploitation sous CP/M 2.2

La partie 7 du Cours Elémentaire montrait comment formater une disquette vierge destinée à la sauvegarde des programmes en BASIC des jeux, ou de CP/M. La partie 10 du Cours Elémentaire montrait comment faire des duplications fidèles de disquettes à l'aide du programme DISCKIT 3 (face 1 de l'une des disquettes système). Ce chapitre concerne la façon de préparer une disquette destinée à recevoir des programmes et des utilitaires de votre choix.

La duplication de la Disquette Système

Il est très important de faire une copie de la disquette système fournie avec votre ordinateur et de garder l'original en lieu sûr. Il vous en coûterait beaucoup pour la remplacer !

Rappelez-vous que chacune des disquettes fournies avec le CPC6128 possède deux faces. Vous avez donc quatre faces au total.

La face 1 est la plus importante car elle contient la copie du système d'exploitation CP/M Plus et un ensemble d'utilitaires servant au traitement des disquettes. La face 2 comprend des fichiers destinés aux programmeurs-assembleurs et la face 3, le Dr. LOGO, les fichiers « Help » et les extensions graphiques GSX (nous y reviendrons plus tard). Quant à la face 4, elle contient le système CP/M2.2 et la version du Dr. LOGO disponible précédemment pour les ordinateurs Schneider CPC664 et CPC464 + DDI1: ces programmes vous sont fournis pour des raisons de compatibilité. Ils ne vous serviront que rarement.

Vous devez conserver ces copies sous forme de « bibliothèque » de programmes. Vous sélectionnerez le programme qui vous intéresse en prenant une des disquettes de cette bibliothèque plutôt qu'en effectuant une copie du dit programme sur une disquette vierge pour le lancer ensuite.

Nous insistons bien sur le fait que les disquettes que vous utiliserez DOIVENT ETRE DES COPIES effectuées à partir des disquettes système fournies avec l'ordinateur.

Si vous prenez une disquette vierge pour y faire une copie, le programme **DISCKIT3** (face 1) la formatera avant d'exécuter la copie.

Introduction au CP/M Plus

Le BASIC Schneider entre en jeu dès la mise sous tension de votre ordinateur. Il se charge des opérations jusqu'à ce que vous lanciez un programme BINaire à partir de l'AMSDOS (ou d'une cassette) ou que vous chargiez le CP/M Plus à l'aide de la commande |**CPM**.

Après chargement de CP/MPlus, le CPC6128 n'a plus besoin de la face 1 de la disquette système, à moins, bien sûr, que vous ne désiriez exploiter l'un des utilitaires stockés sur cette face de la disquette. Seule la disquette d'amorçage du système doit être une disquette système ; par la suite, vous pouvez insérer des disquettes de données à plus grande capacité de stockage.

Pour lancer un programme, il suffit d'insérer la disquette contenant le programme désiré et de taper son nom. Les données utilisées par le programme peuvent se trouver sur la même disquette que celui-ci, ou sur une autre. CP/MPlus autorise la permutation des disquettes tout comme l'AMSDOS. Si plusieurs programmes et quelques utilitaires doivent se trouver sur la même disquette pour des raisons de commodité, utilisez le programme **PIP** sur la face 1 d'une des disquettes système (voir plus loin, dans ce chapitre, et chapitre 5).

Création de profils

L'une des disquettes système fournit un fichier spécial appelé **PROFILE.SUB** contenant une liste de commandes exécutées automatiquement dès le lancement de CP/MPlus. Vous pouvez alors (si ce n'est pas déjà fait) insérer une copie de la face 1 des disquettes système et taper à la suite du message **A**:

```
REN PROFILE.SUB=PROFILE.ENG
```

Vous créez ainsi le fichier **PROFILE.SUB** à partir de **PROFILE.ENG**. Ce profil, qui sera exécuté au prochain lancement de CP/M Plus, contient les commandes:

```
SETKEYS KEYS.CCP
LANGUAGE 3
```

Elles permettent d'adapter les touches du clavier à la frappe des commandes CP/M et convertir certains caractères (pour l'obtention du signe **« £ »**, par exemple, en tapant **[SHIFT]3**.

Après utilisation de la commande **SET KEYS.CCP**, les lignes de commandes CP/M sont éditées au clavier de la même façon qu'en BASIC. Pour plus de détails, consultez la partie 2 du chapitre 5.

Une main tendue

La face 3 d'une des disquettes système comporte un programme spécial appellé « Help », conçu comme un manuel didactique électronique pour les utilitaires CP/MPlus. Pour activer cette fonction, insérez la face 3 et tapez, à la suite de **A**> :

```
HELP
```

Le programme « Help » vous présente alors une série de questions pour vous fournir les informations d'aide requises.

Une ou deux unités?

Au premier chargement de CP/MPlus, le système détecte le nombre d'unités de disquettes reliées à l'ordinateur et l'affiche à la suite du message d'identification. Ce procédé risque d'être faussé si la disquette de la seconde unité n'est pas entièrement insérée.

Tous les messages d'erreur sont affichés sous forme de bannière à la vingt-cinquième ligne de l'écran, les 24 première lignes étant utilisées par les programmes.

Même si votre système ne possède qu'une unité, le message **« Drive is A: »** ou **« Drive is B: »** sera affiché sur la dernière ligne de l'écran: en effet, CP/MPlus vous permet de travailler avec une seule unité physique comme si vous en possédiez deux. Vous pourrez alors alterner entre deux disquettes, et la bannière de message vous indiquera quand insérer la bonne disquette, suivant les besoins. Cette façon de procéder vous évite l'achat d'une seconde unité mais elle exige une permutation fréquente des disquettes; elle se révèle donc fastidieuse et génératrice d'erreurs.

Copie de fichiers d'une disquette sur une autre

Pour copier des fichiers d'une disquette sur une autre, vous disposez d'un utilitaire standard **PIP** (Peripheral Interchange Program/Programme d'Interconnexion aux Périphériques).

Chargez le **PIP** à partir de la face1 et tapez à la suite de **A** >:

```
PIP
```

Un nouveau message, *, vous indique le chargement correct du **PIP**. Normalement, vous copiez des fichiers d'une disquette source (unité **A**) sur une disquette cible (unité**B**). Nous avons vu que dans un système à une seule unité le procédé est le même.

Pour copier un fichier, par exemple **SUBMIT.COM**, tapez à la suite du message *:

```
B:=A:SUBMIT.COM
```

Pour copier la totalité des fichiers d'une disquette source sur une disquette cible, la commande sera:

```
B:=*.*
```

Pour sortir du programme **PIP**, appuyez sur **[RETURN]** à l'apparition du message *.

PIP est un programme très perfectionné; son fonctionnement sera approfondi au chapitre 5.

Une disquette réservée au BASIC

Comme nous l'avons déjà vu, les disquettes système ne servent qu'à l'amorçage du système CP/MPlus. Les disquettes destinées au BASIC peuvent donc ne contenir que des données et offrir une capacité de stockage légèrement supérieure.

La disquette doit être formatée à l'aide du programme **DISCKIT3**. Pour copier des programmes sur cette disquette, vous devez utiliser **PIP** (chargé à partir de la face 1) ou **LOAD** suivi de **SAVE** sous BASIC.

Progiciel en BASIC Schneider

Si vous achetez un programme d'application écrit en BASIC Schneider pour le 6128, son exécution commencera dès la mise sous tension. Il ne vous reste qu'à le recopier sur une disquette de travail.

Progiciel sous CP/M

Le système d'exploitation CP/M vous donne accès à une immense librairie de logiciels, écrits pour des micro-ordinateurs travaillant sous CP/M. La « logique » fondamentale de ces programmes étant déjà définie, la seule chose qui vous reste à faire est de les transférer sur une disquette correcte de votre 6128 et de les informer sur sa manière particulière de gérer l'écran.

Un ensemble de programmes sur une disquette ayant pour objet une application spécifique est appelé un « progiciel ». Ces progiciels sont habituellement conçus pour être utilisés sur un bon nombre d'ordinateurs différents, ayant chacun leur taille d'écran particulière et leur propre système de déplacement de curseur.

Le 6128 dispose d'un « émulateur de terminaux » intégré pour l'exploitation des programmes CP/M Plus, dont les caractéristiques diffèrent des codes de contrôle gérés par le BASIC.

Le progiciel pourra avoir été défini pour le système Schneider ou pour son adaptation au 6128. Le cas échéant, suivez les instructions fournies avec le logiciel pour un protocole Zénith Z19/Z29. S'il n'a pas été prévu de variante pour le système Schneider, le paragraphe « Configuration d'un programme CP/M », un peu plus loin, vous indiquera les commandes de gestion d'écran permettant d'adapter le progiciel au CPC6128. Cette procédure s'effectue normalement en entrant au clavier des codes spécifiques. Encore une fois, suivez les instructions fournies avec le progiciel.

La disquette contenant le logiciel que vous avez acheté doit être utilisable par le système. La majorité des ordinateurs utilisant leur propre format, l'identité de taille des disquettes ne signifie pas qu'elles soient compatibles. Demandez à votre fournisseur la version 3 pouces d' Schneider .

Création d'un progiciel sous CP/M sur disquette

Il est souvent utile d'avoir sur une disquette de progiciel, avec le programme d'application, les utilitaires **SETKEYS.COM** et **SUBMIT.COM** (ainsi que leurs fichiers d'instructions).

PIP peut servir à transférer les fichiers **.COM** et à créer le fichier d'instructions pour **SUBMIT. PIP**, dans ce dernier cas, se révèle être en effet un éditeur ligne à ligne efficace. Par exemple, le fichier **LOGO3.SUB** sur la face 3 aurait pu être créé avec ces commandes:

(insérez la disquette système dans l'unité **A**, face 1 vers le haut) Tapez:

```
PIP
```

(retirez la disquette et insérez la disquette cible), tapez:

```
LOGO3.SUB=CON:
SETKEYS KEYS.DRL
[CONTROL]J LOGO3
[CONTROL]Z
```

La configuration d'un programme CP/M

Le 6128 possède une gamme étendue de codes de référence pour l'adaptation de progiciels sous CP/M. La plupart de ces prologiciels ont besoin d'afficher et de lire des messages n'importe où sur l'écran ainsi que de comprendre les commandes de gestion du curseur.

Si votre logiciel a déjà été adapté pour le système Schneider, ce qui suit ne vous concerne pas.

Adaptation de votre progiciel : les sorties (Output)

La procédure de mise en service d'un progiciel consiste à exécuter un programme spécial (souvent appelé **INSTAL**) qui vous pose un certain nombre de questions concernant les paramètres de l'écran du 6128 s'il ne gère pas un terminal de type Z19/Z29 ou le 6128 en particulier. Inspirez-vous des réponses du tableau ci-dessous (extrait de la partie 15 du chapitre 7).

Codes de contrôle		Hexa	Décimal		Opération effectuée
[BEL]		&07		7	Cloche
[BS]		&08		8	Déplacement curseur-gauche
[LF]		&0A		10	Déplacement curseur-bas
[CR]		&0D		13	Retour chariot
[ESC] A	&1B	&41	27	65	Déplament curseur-haut
[ESC] C	&1B	&43	27	67	Déplacement curseur-droite
[ESC] E	&1B	&45	27	69	Effacement écran
[ESC] H	&1B	&48	27	72	Retour curseur en haut à gauche
[ESC] J	&1B	&4A	27	74	Effacement de l'écran depuis le curseur (inclus) jusqu'à la fin
[ESC] K	&1B	&4B	27	75	Effacement de l'écran depuis le curseur (inclus) jusqu'au bord droit
[ESC] L	&1B	&4C	27	76	Insertion de ligne
[ESC] M	&1B	&4D	27	77	Suppression de ligne
[ESC] N	&1B	&4E	27	78	Effacement du caractère sous le curseur
[ESC] Y	&1B	&59	27	89	Déplacement curseur à une position définie de l'écran. c = colonne + 32, r = ligne + 32
[ESC] d	&1B	&64	27	100	Effacement de l'écran depuis le début jusqu'au curseur (inclus)
[ESC] o	&1B	&6F	27	111	Effacement de l'écran depuis le bord gauche jusqu'au curseur (inclus)
[ESC] p	&1B	&70	27	112	Entrée en vidéo inverse
[ESC] q	&1B	&71	27	113	Sortie de vidéo inverse

Adaptation du progiciel : les entrées (Input)

Les programmes du progiciel doivent avoir la possibilité d'interroger le clavier. A l'exception des déplacements curseur, la plupart des touches du clavier du 6128 ont des valeurs standard. Bien qu'il soit préférable de faire en sorte que le progiciel accepte les valeurs standard par défaut, il est possible d'utiliser SETKEYS pour redéfinir les codes donnés par le clavier.

Malheureusement, il n'y a pas d'uniformité d'un logiciel à l'autre, pour les touches d'activation des fonctions de contrôle par exemple. Les touches de caractères, la barre d'espacement, les touches **[TAB]** et **[RETURN]** sont généralement les mêmes, mais pour l'espace arrière et bien d'autres fonctions les choses se compliquent. En effet, comparez par exemple les codes de déplacement du curseur en début de ligne:

Pour le CP/M: **[CONTROL]B**
Pour Dr.LOGO: **[CONTROL]A**
et pour un autre traitement de texte ce sera **[CONTROL]Q S**

Trois jeux de codes clavier sont fournis en standard. Chaque configuration peut s'obtenir à partir des fichiers stockés sur la face 1 d'une des disquettes système:

```
SETKEYS KEYS.CCP
```

Cette commande envoyée automatiquement par **PROFILE.SUB** définira le clavier correspondant aux commandes CP/M.

Démarrage d'un progiciel CP/M prêt à l'emploi

Normalement, vous n'avez qu'à entrer le nom du progiciel après l'apparition du **A**>. Ainsi, pour lancer un programme de traitement de bulletins de paye appelé **SALAIRE. COM**, tapez simplement :

```
PAYROLL
```

Si vous devez définir une configuration, un fichier **SUB**mit pourra être fourni. Le fichier **LOGO3.SUB**, sur la face 3 d'une des disquette système, en est un exemple. Vous l'appelerez par la commande:

```
SUBMIT LOGO3
```

Vous pourrez voir le contenu de ce fichier en tapant:

```
TYPE LOGO3.SUB
```

Ce qui vous donne:

`SETKEYS KEYS.DRL`	pour changer de clavier
`LOGO3`	pour lancer le programme Dr. LOGO
`SETKEYS KEYS.CCP`	pour revenir au clavier

Démarrage automatique des progiciels CP/M prêts à l'emploi

Le système d'exploitation CP/M Plus permet un démarrage automatique de n'importe quel programme chaque fois que l'on charge le CP/M Plus. Cette fonction est réalisée par l'une des options **PROFILE.SUB**. (Voir la partie 2 du chapitre « AMSDOS et CP/M » pour de plus amples informations).

Introduction aux extensions graphiques GSX

GSX est une extension graphique du système qui permet aux programmes CP/M d'exploiter des graphiques, en plus des textes. Vous pouvez ainsi dessiner des graphiques à barres et à secteurs ainsi que mettre vos en-têtes en évidence en combinant différents styles de caractères en plusieurs tailles. Une reproduction, page suivante, illustre cette fonction. Avec GSX, vous pouvez recevoir vos données sur écran, imprimante ou table traçante.

GSX employé seul ne permet pas de dessiner des graphiques, pas plus que CP/M ne peut effectuer du traitement de texte sans programme d'application spécifique. L'exemple fourni a été obtenu à partir du programme « DR Graph » de Digital Research. GSX, pour sa part, apporte les fonctions de gestion d'écran, d'imprimante et de table traçante permettant de déplacer les programmes d'un ordinateur à l'autre en réduisant au minimum les procédures de réinstallation.

Pour créer une disquette adaptée à l'exploitation de programmes GSX, copiez, à partir de la face 3 de la disquette système, les fichiers **GSX.SYS**, **ASSIGN.SYS** et les gestionnaires de périphérique (ainsi que le programme d'application lui-même) sur une disquette vierge formatée. Le fichier **ASSIGN.SYS** contient un « cocktail » d'un maximum de trois gestionnaires de périphériques de sortie, classés par taille décroissante:

```
21a:ddfxlr7     ;imprimante Epson 7 bit
11a:ddhp7470    ;table traçante
01a:ddmode2     ;écran en mode 2
02a:ddmode1     ;écran en mode 1
03a:ddmode0     ;écran en mode 0
```

Les nombres indiquent au GSX le type de chaque gestionnaire de périphérique (imprimante/table traçante/écran). Les gestionnaires ne peuvent être chargés qu'un par un dans la même zone mémoire, c'est pourquoi GSX doit commencer par le plus important pour pouvoir allouer un espace mémoire suffisant.

Une sélection de gestionnaires est disponible pour les différents modes écran du 6128 et pour les imprimantes standard. Le fichier **DRIVERS.GSX** contient un récapitulatif des gestionnaires fournis avec le 6128. Vous devez le consulter en insérant la face 3 de la disquette système et en tapant à la suite du message **A** >:

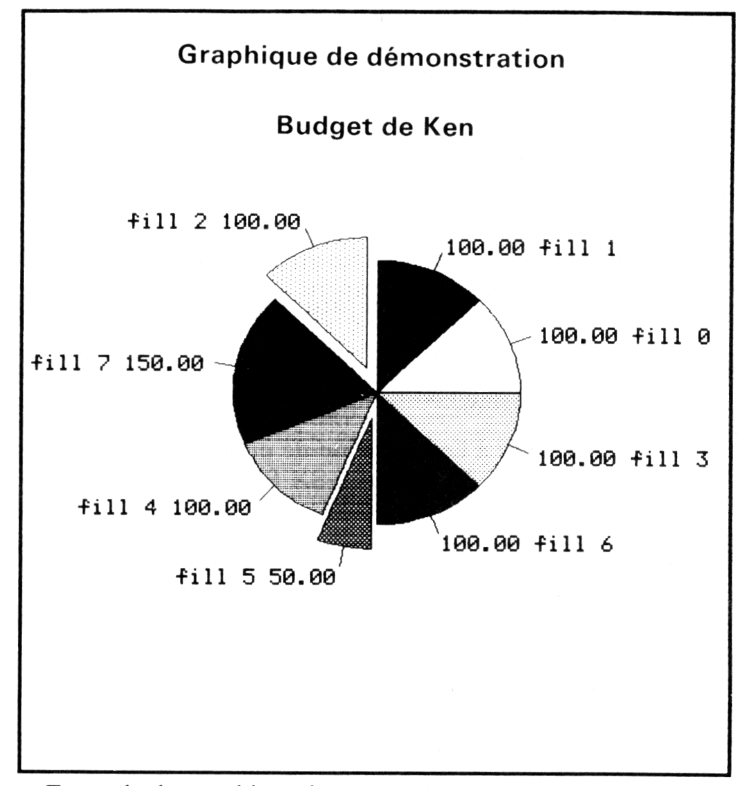
```
TYPE DRIVERS.GSX
```

afin de sélectionner les gestionnaires de périphérique appropriés.

La plupart des programmes d'application comportent un chargeur de GSX; il vous suffit donc de taper le nom du programme à la suite de **A** >. Si votre application ne possède pas de chargeur de GSX installé, copiez le fichier **GENGRAF.COM** (face 3 d'une des disquettes système) sur votre disquette de travail sous GSX et tapez:

```
GENGRAF TONFICHI
```

YOURFILE.COM étant ici l'application non installée. Le fichier **GENGRAF.COM** pourra alors être effacé (commande **ERA**), puisque **YOURFILE.COM** est désormais muni du chargeur de GSX.



Exemple de graphique à secteurs imprimé à partir de GSX

Exploitation sous CP/M2.2

Contrairement au CP/MPlus, le CP/M2.2 (version précédente du système d'exploitation CP/M), très rigoureux quant aux changements de disquettes, exige souvent le rechargement de certaines parties du système d'exploitation à partir d'une disquette système insérée dans l'unité**A**, surtout à la fin de l'exploitation d'un programme, lorsque **A** > réapparaît à l'écran. Si vous ne possédez qu'une seule unité de disquette, vous devez utiliser **FILECOPY** à la place de PIP pour copier vos fichiers.

L'emploi du CP/M2.2 doit se limiter aux logiciels tournant sur les ordinateurs AMSTRAD CPC664 et CPC464 + DDI1, uniquement dans le cas où ceux-ci ne sont pas compatibles avec le CP/MPlus.

ATTENTION : Certains logiciels d'application fonctionnant sous CP/M2.2, sur CPC464 et CPC464/DDI1, contiennent des instructions d'installation spécifiques et ne peuvent pas s'exploiter sous CP/MPlus. Dans ces cas là, vous devrez faire appel au système CP/M2.2 résidant sur la face 4 d'une des disquettes système.

Partie 2 : Les cassettes

Lorsque vous connectez un lecteur de cassettes au système (voir la partie 2 du Cours Elémentaire), un certain nombre de commandes BASIC fonctionnent différemment après l'entrée de la commande | **TAPE**. Cette instruction configure l'ordinateur pour une utilisation avec cassettes et les messages apparaissant à l'écran sont alors différents de ceux vus lors de l'exploitation des disquettes.

REMARQUE : Si les commandes ENROULEMENT RAPIDE et RETOUR ARRIÈRE de votre lecteur sont sous télécommande, vous devez – pour ces opérations – retirer la fiche REM de votre lecteur (télécommande) ou taper | TAPE:CAT pour activer le moteur du lecteur. Appuyez sur **[ESC]** pour revenir au mode télécommande.

Contrairement aux noms de fichiers sur disquettes, les noms de fichiers sur cassettes n'obéissent pas à des règles très strictes quant à leur forme. Ils peuvent comporter jusqu'à 16 caractères incluant des espaces et des signes de ponctuation, ou peuvent même être omis.

La liste ci-après décrit les DIFFERENCES de fonctionnement de ces commandes BASIC. Les commandes elles-mêmes sont décrites dans le chapitre « Liste complète des mots clés du BASIC Schneider CPC6128 ».

CAT

Le message ci-dessous apparaît :

```
Press PLAY Then any key:
```

Vous devez alors appuyer sur le bouton **PLAY** de votre lecteur de cassette puis sur l'une des touches de l'ordinateur. La cassette commence à se dérouler et l'ordinateur charge le fichier à traiter.

Chacun des blocs constituant un fichier va s'afficher, suivi d'un caractère définissant son type :

$ fichier BASIC standard
% fichier BASIC protégé
* fichier ASCII
& fichier binaire

L'ordinateur affiche :

```
Ok
```

...à la fin de la ligne si la lecture du fichier a abouti, indiquant qu'un chargement du fichier aurait été effectué avec succès si la commande adéquate avait été lancée.

La fonction **CAT** n'agit pas sur le programme en cours stocké dans la mémoire.

Si un fichier sur cassette est sauvegardé sans spécification de nom, **CAT** l'affiche comme :

```
Unnamed file
```

Pour arrêter **CAT**, appuyez sur la touche **[ESC]**.

Erreurs de lecture

Si le message ci-dessus indiquant la lecture du fichier n'apparaît pas, ou si vous obtenez :

```
Read error a
```

... ou

```
Read error b
```

Cela signifie que :

1. Votre lecteur de cassettes n'est pas correctement connecté à la prise TAPE de l'ordinateur (voir alors la partie 2 du Cours Elémentaire).

2. Les commandes **VOLUME** ou **LEVEL** de votre lecteur de cassette ne sont pas bien réglées.

3. La cassette est de mauvaise qualité ou usagée.

4. La cassette a été soumise à un champ magnétique provenant d'un haut-parleur, d'un téléviseur, ou d'une autre source.

5. Le type de votre cassette ne convient pas aux systèmes informatiques Schneider.

CHAIN
CHAIN MERGE
LOAD
MERGE
RUN

Vous n'avez pas à spécifier de nom de fichier si vous désirez charger le premier fichier de la cassette. Exemples de commandes :

```
CHAIN  ""
CHAIN  "",100

CHAIN MERGE ""
CHAIN MERGE "",100
CHAIN MERGE "",100,DELETE 30-70
```

```
LOAD ""
LOAD "",&1F40

MERGE ""
```

`run ""` (En maintenant la touche **[CONTROL]** et en appuyant sur la touche **[ENTER]** du clavier numérique, vous exécutez cette commande. S'emploie avec les logiciels sur cassette après l'entrée de | TAPE).

Le BASIC vous répond :

```
Press PLAY then any key:
```

Vous devez alors appuyer sur le bouton **PLAY** de votre lecteur de cassettes puis sur l'une des touches de l'ordinateur. La cassette commence à se dérouler et l'ordinateur charge le fichier à traiter.

Les messages de chargement apparaissent à l'écran :

```
Loading NOMFICH block 1
```

... accompagnés d'autant de numéros de blocs qu'en contient le fichier.

Si le nom de fichier commence par !, les messages ci-dessus sont supprimés et vous n'avez pas à actionner de touche pour charger le fichier (assurez-vous toutefois que le bouton **PLAY** de votre lecteur de cassette est bien enfoncé). Si vos programmes font appel au point d'exclamation ! et doivent aussi tourner sur disquette, le ! est ignoré lors de l'exploitation sur disquette). Le ! n'occupe pas un caractère dans les noms de fichiers sur cassette ou disquette.

A l'abandon de cette commande par la touche **[ESC]**, le message d'erreur ci-dessous s'affiche :

```
Broken in
```

Si le chargement du fichier n'a pas abouti, reportez-vous au paragraphe « Erreurs de lecture ».

ATTENTION : L'interface interne de la disquette occupe une petite partie de la mémoire qui dans certains cas est exploitée par les programmeurs professionnels de logiciels sur cassette destinés au CPC464. Ces cassettes ne fonctionnent pas correctement sur le système 6128 + lecteur de cassettes.

EOF
POS (# 9)

Ces fonctions opèrent de manière identique sur disquette et sur cassette.

INPUT # 9
LINE INPUT # 9
OPENIN et CLOSEIN

Vous n'avez pas à spécifier de nom de fichier pour charger le premier fichier sur cassette. Exemple de commande :

```
OPENIN ""
```

Vous obtenez le message :

```
Press PLAY then any key:
```

Vous devez alors appuyer sur le bouton **PLAY** de votre lecteur de cassettes puis sur l'une des touches grises de l'ordinateur. La bande commence à se dérouler et l'ordinateur charge les 2 premiers Koctets du fichier dans la zone mémoire appelée « tampon de fichiers ». Les informations sont entrées jusqu'à ce que le tampon soit vide. L'ordinateur envoie alors à nouveau un message :

```
Press PLAY then any key:
```

...puis charge les 2 Koctets suivants.

Les messages de chargement s'affichent :

```
Loading NOMFICH block 1
```

...ainsi que les autres numéros de bloc, au fur et à mesure du chargement.

Si le premier caractère du nom de fichier dans la commande **OPENIN** est un point d'exclamation (!), les messages ci-dessus sont supprimés ; vous n'avez à appuyer sur aucune touche pour charger le fichier. (Vous devez alors vous assurer que le bouton **PLAY** de votre lecteur de cassette est bien enfoncé). Si vos programmes font appel au signe ! et doivent également fonctionner sur disquette, le ! est ignoré lors de l'exploitation de la disquette (au cours de la lecture du nom de fichier sur disquette). Le ! n'occupe pas de caractère dans les noms de fichiers sur disquette ou cassette.

Lorsque vous interrompez l'entrée du fichier à l'aide de la touche **[ESC]**, le message suivant s'affiche :

```
Broken in
```

Si le chargement du fichier n'a pas abouti, consultez le paragraphe antérieur « Erreurs de lecture ».

LIST # 9
OPENOUT et CLOSEOUT
PRINT # 9
WRITE # 9

Si vous désirez sauvegarder un fichier sans nom, vous n'avez pas à en spécifier le nom. Exemple de commande :

```
OPENOUT ""
```

Les premiers 2 Koctets du fichier à sauvegarder vont s'inscrire dans la partie de la mémoire appelée « tampon de fichiers ». Lorsque ce tampon est plein, le message ci-dessous apparaît :

```
Press REC and PLAY then any key:
```

Vous devez alors appuyer sur les boutons **RECORD** et **PLAY** de votre lecteur de cassettes puis sur l'une des touches de l'ordinateur. La bande se déroule et l'ordinateur sauvegarde le contenu du tampon. L'ordinateur remplit ensuite à nouveau le tampon, avec les 2 Ko suivants puis envoie le message :

```
Press REC and PLAY then any key:
```

Les 2 Ko suivants sont alors sauvegardés sur la cassette.

Si le tampon est partiellement plein et que la commande **CLOSEOUT** intervient, l'ordinateur sauvegarde sur la cassette ce qui reste dans le tampon, affichant le message :

```
Press REC and PLAY then any key:
```

Le message de sauvegarde s'affiche :

```
Saving NOMFICH block <x>
```

Si le nom de fichier de la commande **OPENOUT** commence par !, les messages ci-dessous n'apparaissent pas et vous n'avez à appuyer sur aucune touche pour sauvegarder le fichier. (Vous devez alors vous assurer que les boutons **RECORD** et **PLAY** de votre lecteur sont bien enfoncés). Si vos programmes font appel à ! et doivent aussi être exploités sur disquette, le ! est ignoré lors de l'exploitation de la disquette (pendant la lecture du nom de fichier sur disquette). Le ! n'occupe pas de caractère dans les noms de fichiers sur disquette ou cassette.

Lorsque vous interrompez la sortie des fichiers en appuyant sur **[ESC]**, le message d'erreur ci-dessous apparaît :

```
Broken in
```

Pour réussir une sauvegarde...

1. Vérifiez que votre lecteur de cassettes est bien connecté à la prise TAPE de l'ordinateur (voir la partie 2 du Cours élémentaire).
2. Vérifiez que les commandes **RECORD** et **LEVEL** de votre lecteur sont correctement réglées.
3. Vérifiez que vos cassettes ne sont pas de mauvaise qualité ou de 120 mn (C120). Il est préférable d'utiliser des cassettes AMSOFT C15.
4. N'exposez vos cassettes à aucun champ magnétique : haut-parleurs, téléviseur, etc...
5. Avant de détruire un programme de la mémoire après sauvegarde, assurez-vous, par la commande **CAT**alog, que la sauvegarde a effectivement abouti.
6. Veillez à l'entretien régulier de votre lecteur de cassettes et à la propreté des têtes de lecture.

SAVE

Si vous désirez sauvegarder un programme comme fichier sans nom, vous n'avez pas à spécifier de nom de fichier. Exemple de commande :

```
SAVE ""
```

Le message ci-dessous s'affiche :

```
Press REC and PLAY then any key:
```

Vous devez alors appuyer sur les boutons **RECORD** et **PLAY** de votre lecteur de cassettes puis sur l'une des touches de votre ordinateur. La bande se déroule et l'ordinateur sauvegarde le programme.

Les messages de sauvegarde apparaissent :

```
Saving NOMFICH block 1
```

...accompagnés d'autant de numéros de blocs que le fichier en contient, jusqu'à la fin de la sauvegarde.

Si le nom de fichier commence par !, les messages ci-dessus n'apparaissent pas et vous n'avez à appuyer sur aucune touche pour sauvegarder le fichier. (Vous devez alors vous assurer que les boutons **RECORD** et **PLAY** de votre lecteur sont bien enfoncés). Si vos programmes font appel à ! et doivent aussi être exploités sur disquette, le ! est ignoré lors de l'exploitation de la disquette (pendant la lecture du nom de fichier sur disquette). Le ! n'occupe pas de caractère dans les noms de fichiers sur disquette ou cassette.

Lorsque vous interrompez cette commande à l'aide de la touche **[ESC]**, le message d'erreur ci-dessous s'affiche :

```
Broken in
```

Reportez-vous au paragraphe « Pour réussir une sauvegarde » plus haut dans ce chapitre.

SPEED WRITE

Cette commande ne fonctionne que sur cassette et peut être exploitée lorsque l'ordinateur fonctionne sur disquette.

Messages d'erreur

Les messages d'erreur 7, 21, 24, 25, 27 et 32 (voir partie 6 du chapitre « Pour information ») s'affichent, le cas échéant, lors du fonctionnement sur cassette.

Commandes externes de l'AMSDOS

Les commandes ci-dessous permettent d'acheminer les entrées/sorties vers la disquette ou la cassette :

| **TAPE** (pouvant se subdiviser en | **TAPE.IN** et | **TAPE OUT**)
| **DISC** (pouvant se subdiviser en | **DISC.IN** et | **DISC.OUT**)

Toutes les commandes externes :

```
|A
|B
|CPM
|DIR
|DRIVE
|ERA
|REN
|USER
```

...sont exploitées sur disquette même si le fonctionnement sur cassette a été sélectionné.

Chapitre 5
Les éléments de l'AMSDOS et du CP/M

Partie 1 : AMSDOS

Sujets abordés :

* Introduction à AMSDOS
* Le catalogue de la disquette
* Changement des disquettes
* Noms et genres des fichiers AMSDOS
* Les en-têtes des fichiers AMSDOS
* Noms de fichiers avec deux unités
* Les jokers
* Un exemple de programme constitué de commandes AMSDOS
* Récapitulatif des commandes d'AMSDOS
* Copie de fichiers
* Guide de référence des messages d'erreur

Introduction

AMSDOS est une extension du BASIC Schneider de votre ordinateur, ajoutant des commandes externes supplémentaires et redéfinissant certaines instructions existantes. Les nouvelles commandes externes sont identifiées par le symbole | (barre).

AMSDOS permet à l'utilisateur de changer librement de disquette (à condition toutefois qu'il n'y ait pas de fichiers ouverts en écriture, auquel cas un message d'erreur apparaît).

Le catalogue de la disquette

Chaque disquette est divisée en 2 parties, une pour le catalogue et une pour les données. Le catalogue est la liste des noms de fichiers que contient la disquette, accompagnée d'un 'plan' de leur positionnement sur la disquette. AMSDOS et CP/M peuvent calculer la taille de chaque fichier après consultation de cette position dans le catalogue. Le calcul de l'espace disponible se fait par l'addition des fichiers existants puis la soustraction du résultat de l'espace disponible du total.

Chaque fois qu'un fichier est lu, il y a d'abord examen de sa position dans le catalogue, déterminant son emplacement sur la disquette. Lorsqu'un nouveau fichier est créé, un espace vacant lui est attribué. Une fois effacé, cet espace est libéré. Le catalogue comptabilise par unité de 1K (un kilo-octet) et peut contenir jusqu'à 64 positions différentes. Les longs fichiers sont dotés d'une position pour chaque groupe de 16K utilisé, cette manœuvre n'apparaissant normalement pas à l'utilisateur.

Changement des disquettes

AMSDOS et CP/M Plus vous permettent de remplacer ou de retirer une disquette à tout moment, en dehors des temps d'accès et si aucun fichier n'est ouvert sur l'unité concernée. Contrairement à CP/M 2.2, AMSDOS autorise cette opération sans procédure d'entrée dans la disquette.

Remplacer une disquette en cours d'écriture risque d'endommager les données qu'elle contient. Lorsque vous remplacez une disquette contenant des fichiers ouverts, ceux-ci sont « abandonnés » dès qu'AMSDOS détecte le remplacement et un message d'erreur s'affiche. Toutes les données à venir sont alors perdues et la dernière position dans le catalogue ne s'inscrit pas sur la disquette. AMSDOS ne détecte le remplacement qu'en lisant le catalogue, c'est-à-dire tous les 16K du fichier (ou lors de toute ouverture ou fermeture d'un fichier). Vous risquez ainsi de perdre 16K de données lorsque vous remplacez une disquette contenant des fichiers ouverts.

Noms et genres des fichiers AMSDOS

Il est d'usage d'ajouter une extension aux noms des fichiers disques. Cette extension, simple convention indiquant le type du fichier, ne joue aucun rôle précis. Certains programmes n'acceptent toutefois un fichier que suivi d'une extension particulière. AMSDOS admet n'importe quel type de nom mais cherche de préférence les fichiers d'un certain type, sauf spécification particulière. (Voir « Les en-têtes des fichiers AMSDOS »).

Construction des noms de fichiers

Le nom est constitué de deux parties séparées par un point, la première partie de 8 lettres maximum et la seconde, de 3. Ainsi, « **ROINTIME.DEM** », « **DISC.BAS** » et « **DISCKIT.COM** » sont des noms de fichiers corrects.

La deuxième partie représente l'indicateur de type. Les noms de fichiers et leurs indicateurs peuvent être composés d'un mélange de lettres et de nombres mais ne peuvent contenir ni espaces blancs, ni signes de ponctuation. Les indicateurs les plus communs sont :

.<espace> Type non spécifié. Peut être un fichier de données créé par l'instruction **OPENOUT** « <nomfich> » (nom de fichier) ou un programme BASIC sauvegardé en utilisant une commande AMSDOS du style **SAVE** « <nomfich> »,**A**.

.BAS Programme BASIC sauvegardé à l'aide de commandes AMSDOS du style **SAVE** « <nomfich> »,**P** ou **SAVE** « <nomfich>.**BAS** »,**A**.

.BIN Programme ou zone de mémoire sauvegardée par une commande AMSDOS du style **SAVE** « <nomfich> »,**B** (<paramètres binaires>).

.BAK Ancienne version d'un fichier sauvegardée lors de la configuration d'une nouvelle version par AMSDOS ou un utilitaire. L'utilisateur est ainsi à même de réutiliser l'ancienne version (**BAcK**-up) si besoin est.

.COM C'est une commande. Tous les utilitaires du CP/M ont cette extension.

.SUB Fichier d'instruction du programme **SUBMIT** sous CP/M.

Les en-têtes des fichiers AMSDOS

AMSDOS dote automatiquement chaque fichier d'un indicateur de type. Si les indicateurs (par défaut) ne vous satisfont pas, vous avez la possibilité d'en spécifier un à votre convenance. L'indicateur de type peut correspondre pour le système à un certain en-tête. Les programmes BASIC AMSDOS, les programmes compilés et les fichiers binaires sont sauvegardés sur disquette avec un en-tête (le même que sur une cassette) permettant à la commande AMSDOS :

LOAD « <nomfich> »

...de les reconnaître et d'agir en conséquence. Si **LOAD** ne trouve pas d'en-tête, il considère avoir affaire à un fichier en codes ASCII.

Lors du chargement d'un fichier dépourvu d'indicateur de type, la commande **LOAD** AMSDOS, indépendamment du contenu de l'en-tête, cherche d'abord ce fichier sous le type :

. <espace>

...puis, si elle ne trouve rien :

.BAS

...et enfin :

.BIN

L'utilisateur a ainsi le loisir d'abréger ses noms de fichiers en omettant, le plus souvent, de spécifier le type.

Un fichier de données sur disquette rempli après ouverture à l'aide de la commande **OPENOUT**, n'aura pas d'en-tête, et son contenu, venu des commandes **BASIC WRITE**, **PRINT**, **LIST**, sera constitué de codes ASCII. En l'absence d'indicateur, la commande **OPENIN** cherchera un fichier de ce type.

Noms de fichiers avec deux unités

Dans le cas d'un système à deux unités de diquettes, une FD-1 étant connectée à l'ordinateur, les fichiers peuvent se trouver sur l'une ou l'autre des deux unités. L'ordinateur n'allant pas chercher sur les deux, vous devrez spécifier l'unité concernée. Pour cela, vous disposez soit de | **A**, soit de | **B**, soit encore de | **DRIVE** suivi d'un nom de fichier normal. Vous pouvez supplanter le choix de l'unité par défaut en adjoignant un préfixe (**A:** ou **B:**) au nom.

Ainsi :

```
|B
SAVE "PROG.BAS"
|A
```

...et :

```
|A
SAVE "B:PROG.BAS"
```

...sauvegardent le programme sur l'unité B.

De même, vous pouvez supplanter l'affectation numéro **USER** (les numéros d'utilisation permettent le découpage du répertoire) par défaut en spécifiant un numéro d'utilisateur, compris entre 0 et 15, qui servira de préfixe au nom de fichier. Ainsi, les commandes :

```
LOAD "15:PROG.BAS"
```

...et :

```
SAVE "15:PROG.BAS"
```

...chargent et sauvegardent le programme sur la section de la disquette affectée à l'utilisateur 15, quel que soit le numéro par défaut (voir commande | **USER**, dans la suite du manuel).

Enfin, il est possible de supplanter à la fois la valeur par défaut de l'utilisateur (**USER**) et celle de l'unité (**DRIVE**) (dans cet ordre) en les indiquant dans le préfixe du nom de fichier, comme par exemple :

```
RUN "15B:PROG.BAS"
```

Les jokers

On a souvent besoin d'opérer (copie, effacement, etc.) sur plusieurs fichiers à la fois. Lorsqu'un nom de fichier est spécifié pour telle ou telle opération, le système recherche le nom donné dans le catalogue. Il est possible dans certains cas de demander à ce que l'opération soit effectuée sur plusieurs fichiers ayant des lettres communes, les autres lettres étant alors remplacées par des ? (signifiant 'ne pas tenir compte de'). On peut abréger une série de ? en la remplaçant par le symbole *. Ainsi, **FRED**.* est le raccourci de **FRED.???** et **F*.BAS** celui de **F???????.BAS**.

Dans cet esprit, *.* signifie 'n'importe quel fichier'.

Exemples :

Catalogue	Convient à *.BAS	Convient à FRED?.BAS	Convient à F*.BAS
BERT.BAS	BERT.BAS		
FRED1.BAS	FRED1.BAS	FRED1.BAS	FRED1.BAS
FRED2.BAS	FRED2.BAS	FRED.BAS	FRED2.BAS
FRED3.BAK			FRED3.BAK
FRED3.BAS	FRED3.BAS	FRED3.BAS	FRED3.BAS
FINAL.BAS	FINAL.BAS		FINAL.BAS

Exemples d'utilisation de commandes d'AMSDOS dans un programme

Pour vous aider à bien comprendre les commandes d'AMSDOS, nous vous conseillons de bien regarder ces exemples et même de revenir sur certaines parties au moment où nous nous y réfèrerons dans la suite du chapitre. Ne faites pas fonctionner ces programmes avec l'original de vos disquettes système CP/M. Utilisez une disquette de travail ou une copie sur lesquelles ils pourront écrire en toute sécurité.

Sauvegarde de variables et recopie d'écran

Le programme ci-dessous inscrit des données sur la disquette ; vous devez donc insérer dans l'unité une disquette vierge ou une disquette de travail. Le programme dessine le chapeau anglais et recopie la totalité de l'écran sur la disquette.

```
10 dumpfile$="flagdumo.srn"
20 MODE 1:BORDER 0
30 DIM colour(2)
40 FOR i=0 TO 2
50  READ colour(i): REM charge les No de couleurs en <DATA>
60  INK i,colour(i)
70 NEXT
80 ON ERROR GOTO 430
90 OPENIN "param.dat" ' teste si le fichier existe
100 CLOSEIN:ON ERROR GOTO 0
110 IF errnum=32 AND DERR=146 THEN CLS:GOTO 160 ' le fichier
 n'existe pas
120 CURSOR 1:PRINT "Do you want to overwrite old file? Y/N "
;
130 a$=INKEY$:ON INSTR(" YN",UPPER$(a$)) GOTO 130,150,140:GO
TO 130
140 PRINT a$:PRINT "Program abandoned":END
150 PRINT a$:CURSOR 0
160 OPENOUT "param.dat"
170 WRITE #9,dumpfile$,1: REM sauvegarde le nom du fichier e
t son mode
180 FOR i=0 TO 2
190 WRITE #9,colour(i): REM sauvegarde les couleurs
200 NEXT i
210 CLOSEOUT
220 CLS
230 gp=1:GRAPHICS PEN gp:w=125
240 x=-65:a=240:y=400:b=-150:GOSUB 400
```

```
250 v=0:b=150:GOSUB 400
260 x=575:a=-240:y=400:b=-150:GOSUB 400
270 v=0:b=150:GOSUB 400
280 gp=2:GRAPHICS PEN gp:w=40
290 a=240:x=-40:y=400:b=-150:GOSUB 400
300 x=0:y=0:b=150:GOSUB 400
310 a=-240:x=640:y=0:b=150:GOSUB 400
320 x=600:y=400:b=-150:GOSUB 400
330 ORIGIN 0,0,256,380,0,400:CLG 1
340 ORIGIN 0,0,0,640,150,250:CLG 1
350 ORIGIN 0,0,280,352,0,400:CLG 2
360 ORIGIN 0,0,0,640,168,230:CLG 2
370 SAVE dumpfile$,b,&C000,&4000
380 DATA 2,26,6
390 END
400 MOVE x,y:DRAWR a,b:DRAWR w,0:DRAWR -a,-b
410 MOVE x+a/2+w/2,y+b/2:FILL gp
420 RETURN
430 errnum=ERR:RESUME NEXT
run
```

Les indicateurs **.DAT** et **.SRN** n'ont d'autre rôle que de nous rappeler ce que contient le fichier. Le fichier **PARAM.DAT** contient des paramètres codés en ASCII, donc sans en-tête, alors que **FLAGDUMP.SRN** est un fichier AMSDOS en code binaire, avec en-tête.

Vous remarquerez que le programme essaie d'abord de lire le fichier **PARAM.DAT** afin de s'assurer qu'il existe avant d'y inscrire des données. S'il n'existe pas, le BASIC signale une erreur qui est capturée par le programme et l'exécution se déroule sans interruption. S'il existe, le programme vous propose d'écraser le fichier existant.

Les particularités de l'écran que l'on va copier, (son mode, la palette de couleurs et le nom du fichier contenant les données) sont copiées sur un fichier. Voici qui illustre l'utilisation d'un fichier de données pour la sauvegarde **WRITE** des variables dans un programme (**dumpfile$**) et de constantes (1), pour réutilisation dans un autre programme.

Restitution d'écran

Cet exemple (**EX2.BAS**) est le programme type de restitution d'une copie d'écran faisant appel au fichier des paramètres pour définir son action. Vous remarquerez la façon d'appeler (**INPUT**) les variables se trouvant dans un fichier avec variation automatique de longueur de fichier par la fonction **EOF** (End Of File = Fin de fichier). Il est important de sauvegarder les paramètres de l'écran, pour éliminer tout risque de brouillage des résultats. La commande **MODE** et le blocage du défilement continu de l'image accomplissent cette sauvegarde.

```
10 DIM colour(15): REM provision for 16 colours
20 OPENIN "param.dat"
30 INPUT #9,filename$,screenmode
40 i=0
50 WHILE NOT EOF
60  INPUT #9,colour(i)
70  INK i,colour(i)
80  i=i+1
90 WEND
100 CLOSEIN
110 MODE screenmode:BORDER 0
120 LOAD filename$
run
```

Sommaire des commandes externes d'AMSDOS

| A

|A

COMMANDE : Dirige les entrées/sorties vers l'unité A. Equivalent à | **DRIVE** avec A pour paramètre. (L'unité intégrée est l'unité A.)

| B

| **B**

COMMANDE : Dirige les entrées/sorties vers l'unité B. Equivaut à | **DRIVE** avec B pour paramètre. (L'unité intégrée est l'unité A.)

| CPM

| **CPM**

COMMANDE : Charge et initialise le système d'exploitation contenu dans la disquette système. Les systèmes fournis avec l'ordinateur sont les CP/M 2.2 et CP/M Plus.

Cette opération échoue si l'unité ne contient pas de disquette système. La copie de la face 1 charge CP/M Plus et celle de la face 4, CP/M 2.2.

| DIR

| **DIR** [,<chaîne alphanumérique>]

```
|DIR,"*.BAS"
```

COMMANDE : Affiche le catalogue de la disquette (façon CP/M) et l'espace disponible. L'absence de la chaîne équivaut à *.*.

| DISC

| **DISC**

COMMANDE : Equivaut aux commandes | **DISC.IN** et | **DISC.OUT** réunies.

| DISC.IN

| **DISC.IN**

COMMANDE : Force la lecture des données de la disquette.

| DISC.OUT

| **DISC.OUT**

COMMANDE : Force l'écriture sur la disquette.

| DRIVE

| **DRIVE,**<chaîne alphanumérique>

```
|DRIVE,"A"
```

COMMANDE : Désigne une des unités. Cette commande avorte si AMSDOS est incapable de lire la disquette dans l'unité désignée.

| ERA

| **ERA,**<chaîne alphanumérique>

```
|ERA,"*.BAK"
```

COMMANDE : Efface tous les fichiers qui correspondent à ce nom de fichier et ne sont pas en Ecriture seulement. Les jockers sont permis.

| REN

| **REN,**<chaîne alphanumérique>,<chaîne alphanumérique>

```
|REN,"NOUVEAU.BAS","ANCIEN.BAS"
```

COMMANDE : Donne un nouveau nom de fichier. Il ne doit pas exister de fichier portant déjà ce nom. Les jockers ne sont pas autorisés.

Le paramètre **USER** (voir | **USER**) peut être spécifié dans la <chaîne alphanumérique> pour supplanter toutes les valeurs par défaut. Ainsi, la commande | **REN,« Ø:NOUVEAU.BAS »,« 15:ANCIEN.BAS »** rebaptise le fichier de l'utilisateur . 15, **« ANCIEN.BAS »**, et l'appelle **« NOUVEAU.BAS »** en l'affectant à l'utilisateur 0, sans tenir compte des valeurs par défaut ou définies précédemment.

| TAPE

| **TAPE**

COMMANDE : Equivaut aux commandes | **TAPE.IN** et | **TAPE.OUT** réunies. Utilisée dans le cas où un lecteur de cassettes est raccordé à l'ordinateur.

| TAPE.IN

| **TAPE.IN**

COMMANDE : Lit des données sur un lecteur de cassettes connecté à l'ordinateur.

| TAPE.OUT

| **TAPE.OUT**

COMMANDE : Envoie des données vers un lecteur de cassettes connecté à l'ordinateur.

| USER

| **USER,** <nombre entier>

```
|USER,3
```

COMMANDE : Détermine la section du catalogue (comprise entre 0 et 15) sur laquelle vont s'effectuer les accès disque.

Un fichier peut être transféré d'un numéro d'utilisateur à l'autre, à l'aide d'un | **REN**. Ainsi, | **REN,« 15:EXEMPLE.BAS »,« Ø:EXEMPLE.BAS »** transfère un fichier de l'utilisateur 0 à l'utilisateur 15, sans modifier le nom du fichier (**EXEMPLE.BAS**).

Copie de fichiers entre disquettes

Fichiers AMSDOS à en-tête

Il est possible de copier ces fichiers sous CP/M, en utilisant **PIP** (voir ce chapitre, partie 2). N'importe quel fichier créé sous AMSDOS comporte une zone d'en-tête (voir « Les en-têtes d'AMSDOS ») et peut être copié intégralement, de disquette à disquette et de cassette à disquette - toutefois, son contenu est en général incompréhensible par CP/M.

Fichiers ASCII

Les fichiers créés sous AMSDOS sans zone d'en-tête sont généralement écrits en codes ASCII et sont compris par AMSDOS et CP/M. Ils peuvent donc s'échanger librement entre AMSDOS et CP/M.

Fichiers en lecture seulement

CP/M permet de limiter l'accès à un fichier en lecture seulement, et/ou selon son état dans le catalogue. De tels attributs sont accordés et supprimés par CP/M tout en étant respectés par AMSDOS. Pour plus de détails, voir la partie 2 de ce chapitre (utilitaire SET).

Copie de fichiers entre disquettes et cassettes

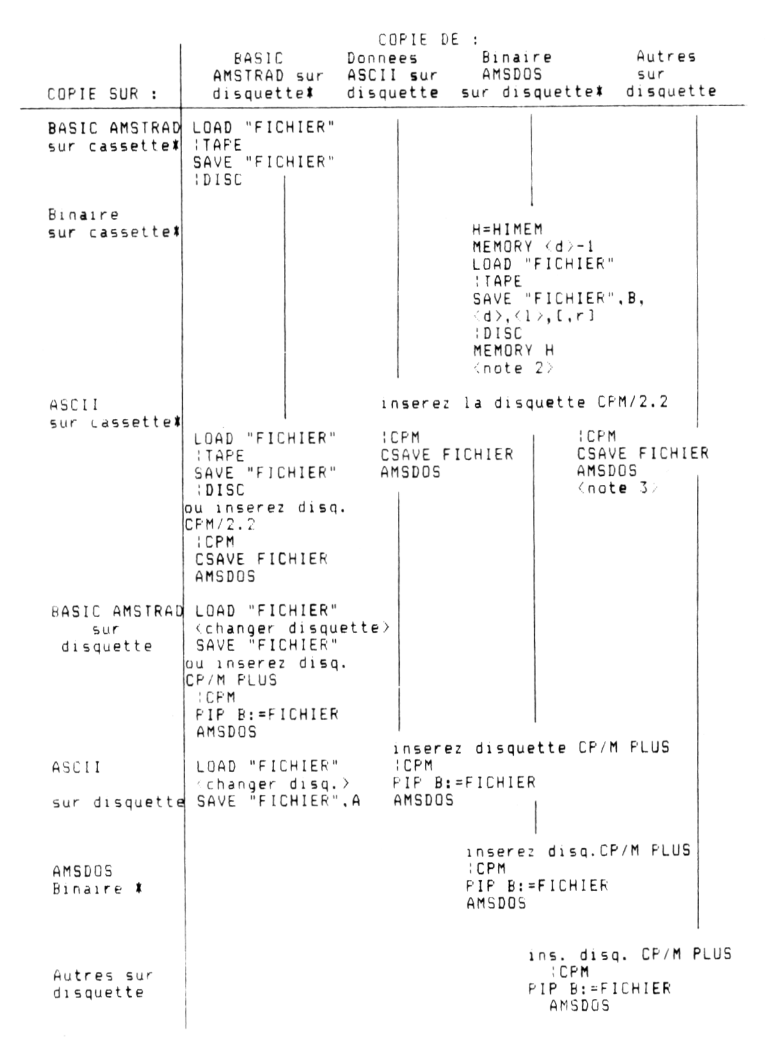
Le CP/M Plus ne permet pas la copie de fichiers entre disquettes et cassettes. Vous devez pour cela faire appel au CP/M 2.2 (face 4 d'une des disquettes système). Les utilitaires **CLOAD** et **SAVE** du CP/M 2.2 vous permettront d'effectuer ce type de copie (reportez-vous aux tableaux pour leur utilisation).

Les procédures de copie de fichiers

Les tableaux qui suivent montrent la façon de copier n'importe quel fichier entre disquettes et cassettes (le cas échéant). Ils sont destinés à un système à une seule unité. Il est absolument impossible de copier un programme BASIC protégé ou un fichier binaire (un jeu écrit en langage machine, par exemple) en provenance et à destination d'une cassette si les adresses de chargement sont inconnues. Vous trouverez plus de détails sur **PIP**, **CLOAD** et **CSAVE** à la partie 2 de ce chapitre.

Voir les tableaux des deux pages suivantes.

COPIE SUR :	COPIE DE : BASIC AMSTRAD sur cassette *	 Donnees ASCII sur cassette*	 Binaire sur cassette*
BASIC AMSTRAD	:TAPE LOAD "FICHIER" <changer cassette> SAVE "FICHIER" :DISC		
Binaire sur cassette*			H=HIMEM :TAPE MEMORY <d>-1 LOAD "FICHIER" <changer cassette> SAVE "FICHIER",B, <d>,<l>,[,r] :DISC MEMORY H <note)
ASCII sur cassette*	:TAPE LOAD "FICHIER" <changer cassette> SAVE "FICHIER",A :DISC	inserez la disquette CP/M 2.2 :CPM CLOAD "FICHIER",TEMP <changer cassette> CSAVE TEMP,"FICHIER" ERA TEMP AMSDOS <note 1>	
BASIC AMSTRAD sur disquette	:TAPE LOAD "FICHIER" :DISC SAVE "FICHIER"		
ASCII sur disquette	:TAPE LOAD "FICHIER" :DISC SAVE "FICHIER",A	inserez la disquette CP/M 2.2 :CPM CLOAD "FICHIER" AMSDOS	
Binaire sur disquette *			H=HIMEM :TAPE MEMORY <d>-1 LOAD "FICHIER" :DISC SAVE "FICHIER",B, <d>,<l>,[,<r>] MEMORY H <note 2>



COPIE SUR :	COPIE DE : BASIC AMSTRAD sur disquette*	Donnees ASCII sur disquette	Binaire AMSDOS sur disquette*	Autres sur disquette
BASIC AMSTRAD sur cassette*	LOAD "FICHIER" :TAPE SAVE "FICHIER" :DISC			
Binaire sur cassette*			H=HIMEM MEMORY <d>-1 LOAD "FICHIER" :TAPE SAVE "FICHIER",B, <d>,<l>,[,r] :DISC MEMORY H <note 2>	
ASCII sur cassette*	LOAD "FICHIER" :TAPE SAVE "FICHIER" :DISC ou inserez disq. CPM/2.2 :CPM CSAVE FICHIER AMSDOS	inserez la disquette CPM/2.2 :CPM CSAVE FICHIER AMSDOS		:CPM CSAVE FICHIER AMSDOS <note 3>
BASIC AMSTRAD sur disquette	LOAD "FICHIER" <changer disquette> SAVE "FICHIER" ou inserez disq. CP/M PLUS :CPM PIP B:=FICHIER AMSDOS			
ASCII sur disquette	LOAD "FICHIER" <changer disq.> SAVE "FICHIER",A	inserez disquette CP/M PLUS :CPM PIP B:=FICHIER AMSDOS		
AMSDOS Binaire *			inserez disq.CP/M PLUS :CPM PIP B:=FICHIER AMSDOS	
Autres sur disquette				ins. disq. CP/M PLUS :CPM PIP B:=FICHIER AMSDOS

* Fichier avec en-tête

<note 1> : requiert de l'espace disque libre pour y loger le programme de temporisation « **TEMP** ».

<note 2> : <d> est l'adresse de départ du dossier, <l> sa longueur et <r> l'adresse de lancement facultative.

<note 3> : la copie est inutilisable directement par le BASIC. Cependant cette option est un moyen très rentable de transfert et de sauvegarde de fichiers. Le fichier peut être ensuite récupéré par la disquette au moyen de **CLOAD« FICHIER »** (sous CP/M 2.2).

Guide des messages d'erreur d'AMSDOS

Lorsque, pour une raison quelconque, AMSDOS ne peut exécuter une commande, il affiche un message d'erreur à l'écran. En cas de problème matériel, le message est suivi de la question :

Retry, Ignore or Cancel ?

R : la commande est reexécutée après que l'utilisateur ait pallié au problème.

I : l'ordinateur continue son travail comme si de rien n'était, amenant souvent des résultats inattendus et probablement erronés.

C : l'opération est annulée, ce qui entraîne souvent un autre message d'erreur.

Signification des messages

Unknown command

La commande n'a pas été écrite correctement.

Bad command

La commande ne peut être exécutée. Erreur de syntaxe ou mauvaise configuration matérielle.

<nomfich> **already exists**

L'utilisateur essaye de renommer un fichier avec un nom déjà utilisé sur la disquette.

<nomfich> **not found**

Le fichier n'existe pas.

Drive <unité> : **directory full**

Plus de position dans le catalogue.

Drive <unité> : **disc full**

Plus de place sur la disquette.

Drive <unité> : **disc changed, closing** <nomfich>

La disquette a été enlevée avec des fichiers laissés ouverts.

<nomfich> **is ready only**

Le fichier est limité en lecture et ne peut être traité. Cet attribut est accordé ou supprimé sous CP/M uniquement.

Drive <unité> : **disc missing**

Disquette absente du lecteur, ou mal mise en place (ne tournant donc pas). Il est alors recommandé de l'éjecter et de la remettre à nouveau, puis de taper **R**.

Drive <unité> : **disc is write protected**

On a essayé d'écrire sur une disquette dont le trou de protection en écriture est ouvert. La sortir, obturer le trou de protection, la réinsérer et taper R.

Drive <unité> : **read fail**

Problème matériel. Il est recommandé d'éjecter la disquette puis de la remettre en place. Taper **R**.

Drive <unité> : **write fail**

Comme précédemment.

Failed to load CP/M

Erreur de lecture au chargement du CP/M (| **CPM**) ou disquette système invalide. Ou bien votre disquette ne contient pas de CP/M. Si vous tentez de charger le CP/M à partir d'une disquette de données, vous obtenez le message « read fail ».

Partie 2 : CP/M

CP/M Plus

Sujets abordés :

* Introduction au CP/M Plus
* Chargement du CP/M Plus
* Le Mode Direct
* Les programmes transitoires
* Gestion des périphériques

Le CP/M Plus est fourni avec le 6128. C'est un système d'exploitation de disquettes qui vous permettra d'utiliser toute la puissance de votre ordinateur. Ses 128Ko de mémoire RAM sont pleinement exploités, laissant plus de 61Ko à l'utilisateur pour ses programmes. CP/M intègre un accès sélectif aux fichiers de données et le CP/M 6128 apporte les avantages d'un émulateur de terminaux (postes VDU) perfectionné. Implanté sur de nombreux ordinateurs, CP/M a suscité le développement de milliers de logiciels d'application et accumulé une expérience et des connaissances énormes, sur lesquelles vous pourrez vous appuyer.

Vous trouverez une documentation complète sur le CP/M Plus, des indications pour l'écriture de vos propres programmes et des informations concernant l'implantation CP/M Plus d' Schneider , dans le guide de CP/M Plus (SOFT 971).

Introduction

Le système d'exploitation CP/M sert d'interface avec le mécanisme de l'unité, vous permettant de communiquer avec l'ordinateur et de manipuler ses fichiers et périphériques.

Des commandes spéciales (et des programmes sur disquette appelés utilitaires) vous aident à réaliser votre principale tâche : l'exploitation de vos programmes d'applications avec vos propres données.

Il va sans dire que vous pouvez devenir expert en matière d'utilisation du CP/M et de fonctionnement des différents utilitaires : cela vous sortira d'affaire plus d'une fois. Cependant, la plupart des utilisateurs peuvent se contenter des connaissances de base permettant de démarrer. La suite de ce chapitre va leur permettre de découvrir les fonctions du système et tous ses avantages, sans que l'essentiel soit enseveli sous un excès de détails.

Le mode direct est l'intermédiaire principal à votre disposition. On l'identifie par la présence des indicateurs d'attente **A**> ou **B**> à l'écran. Certaines commandes, dites résidentes, sont accessibles directement, à l'inverse de la plupart des fonctions, obtenues en chargeant en mémoire des programmes, dits transitoires, pour le temps de leur utilisation uniquement.

A côté des messages d'erreur standard du CP/M, le système génère ses propres messages. Ils se distinguent des autres car ils apparaissent dans la bannière de l'écran.

CP/M Plus sur la disquette

La plus grande partie du CP/M Plus réside dans un fichier spécial de type « **.EMS*51** » qui se trouve sur la face 1 d'une des disquettes système. Le chargement en mémoire du CP/M à partir de ce fichier s'effectue en deux temps.

La commande AMSDOS |**CPM** charge d'abord le premier secteur de la piste 0. Ce secteur de la disquette système contient un programme qui charge le fichier **.EMS** en mémoire. Le reste des pistes système demeure inutilisé.

Création de fichiers PROFILE

Lors du chargement du CP/M Plus, les commandes du fichier **PROFILE.SUB** (s'il se trouve sur la disquette) sont exécutées. Cette particularité permet de redéfinir le clavier, personnaliser le mode d'affichage de l'écran, initialiser l'imprimante et même effectuer le lancement automatique d'un programme d'application. Nous avons vu au chapitre 4 comment renommer le fichier PROFILE résidant sur la face 1 de la disquette système afin qu'il soit exécuté lors du chargement.

Pendant l'exécution du fichier PROFILE, un petit fichier provisoire est ouvert sur la disquette qui ne doit donc pas avoir été protégé en écriture. C'est pourquoi la disquette système ne peut pas contenir de fichier PROFILE exécutable.

Les fichiers PROFILE peuvent s'obtenir à l'aide d'un traitement de texte, d'un éditeur de texte (**ED.COM**, par exemple) ou même à partir du BASIC. Ainsi, le petit programme en BASIC ci-dessous pourrait générer le fichier **PROFILE.SUB**.

```
10 OPENOUT "PROFILE.SUB"
20 PRINT #9,"SETKEYS KEYS.CCP"
30 PRINT #9,"LANGUAGE 3"
40 CLOSEOUT
```

Codes de contrôle de la console

L'univers CP/M, c'est aussi un ensemble de codes clavier spéciaux pour contrôler le déroulement des programmes. Ces codes remplacent le **[ESC]** du BASIC et les touches de déplacement du curseur du BASIC Schneider . Les codes de contrôle ci-dessous sont affectés après lancement de la commande :

Les codes de contrôle ci-dessous sont affectés après lancement de la commande :

```
SETKEYS KEYS.CCP
```

Le programme transitoire **SETKEYS.COM** et le fichier de commande **KEYS.CCP** se trouvent sur la face 1 d'une des disquettes système.

Code de contrôle	Touche	Action
[CONTROL]A	⇦	Déplace le curseur d'un caractère vers la gauche
[CONTROL]B	[CONTROL]⇦ ou [CONTROL]⇨	Déplace le curseur jusqu'au début de la ligne ou, s'il s'y trouve déjà, en fin de ligne.
[CONTROL]C	[CONTROL][ESC]	Abandon
[CONTROL]E	[CONTROL][RETURN]	Retour chariot physique
[CONTROL]F	⇨	Déplace le curseur d'un caractère vers la droite
[CONTROL]G	[CLR]	Efface le caractère situé sous le curseur
[CONTROL]H	[DEL]	Effacement du caractère précédant le curseur
[CONTROL]I	[TAB]	Déplace le curseur jusqu'à la tabulation suivante
[CONTROL]J		Validation de la ligne de commande

[CONTROL]K	**[CONTROL][CLR]**	Efface les caractères jusqu'à la fin de la ligne
[CONTROL]M	**[RETURN]** ou **[ENTER]**	Validation de la ligne de commande
[CONTROL]P		Basculement en mode copie d'écran : toute sortie vers l'écran sera automatiquement imprimée ; une deuxième commande identique permet de revenir au mode normal
[CONTROL]Q		Revalide la sortie des données vers l'écran
[CONTROL]R	**[CONTROL][ENTER]**	Retape une ligne de commande
[CONTROL]S	**[ESC]**	Suspend momentanément la sortie des donnés vers l'écran en provenance du CP/M. Voir **[CONTROL]Q** pour reprendre
[CONTROL]U		Supprime une ligne
[CONTROL]W	**[COPY]**	Rappelle la ligne de commande la plus récemment tapée
[CONTROL]X	**[CONTROL][DEL]**	Efface le début de la ligne jusqu'au curseur
[CONTROL]Z		Fin de texte

Les noms de fichiers

Beaucoup de commandes ont pour paramètres des noms de fichiers contenant, le cas échéant, des jockers (voir le paragraphe « jockers » dans la partie 1 du chapitre). Tous les noms de fichiers sont automatiquement mis en majuscules.

Les commandes du mode direct ne requièrent pas le placement de guillemets de part et d'autre du nom de fichier.

Rappelez-vous que les noms de fichiers peuvent être précédés de **A:** ou **B:** pour demander à CP/M d'utiliser une unité particulière.

Voici la syntaxe d'une commande CP/M type :

```
TYPE KEYS.CCP
```

TYPE est la fonction requise : « affichage à l'écran », et **KEYS.CCP** est le nom du fichier que l'on désire voir apparaître.

L'aiguillage des unités

Il est possible d'aiguiller le système de sélection des unités vers A ou B en tapant **A:** ou **B**: après le symbole **B**> ou **A**> (qui indique l'unité en cours). L'ajout du préfixe **A:** ou **B**: au nom de fichier écrase la valeur de l'unité par défaut mais ne la redéfinit pas.

Les commandes du mode direct

Ces commandes peuvent être tapées à l'apparition des symboles **A** > ou **B** >. Elles peuvent être abrégées. Les fonctions ci-dessous sont intégrées, mais il existe plusieurs commandes transitoires, plus complexes, portant le même nom.

La commande DIR

DIR affiche le catalogue de la disquette. Les fichiers apparaissent selon leur ordre d'entrée sur le répertoire de la disquette. Les jockers sont permis. Les fichiers à attribut « SYS » ne sont pas permis.

DIR	donne la liste de tous les fichiers de l'unité par défaut
DIR B:	donne la liste de tous les fichiers de l'unité B
DIR *.BAS	donne la liste des fichiers de type **.BAS**
DIR B:*.BAS	donne la liste des fichiers de type **.BAS** de l'unité B
DIR PIP.COM	seul **PIP.COM** apparaît sur la liste (s'il existe)

La commande DIRS ou DIRSYS

DIRSYS, ou **DIRS**, ne liste que les fichiers du catalogue possédant « **SYS** » comme attribut. A cette particularité près, cette commande fonctionne comme la commande **DIR**. L'attribut **SYS** est décrit plus loin.

La commande ERA ou ERASE

ERA vient de l'anglais **ERA**se qui veut dire effacer. Elle sert donc à effacer un ou plusieurs fichiers du catalogue. En fait, la position sur le répertoire est effacée, les données restant en place jusqu'à ce qu'un autre fichier ne les recouvre. Elles ne peuvent toutefois jamais être récupérées. Bien que les jockers soient permis, la spécification de *.* provoque l'affichage d'une demande de confirmation avant que **ERA** n'efface effectivement l'ensemble des fichiers. **ERA** ne mentionne pas le nom des fichiers qu'elle efface. Si l'un des fichiers à effacer est de type 'lecture/seulement', (voir **STAT** plus loin), la commande n'aboutit pas.

ERA PIP.COM	effacera le fichier **PIP.COM**
ERA B:PIP.COM	effacera le fichier **PIP.COM** de l'unité B
ERA *.BAS	effacera tous les fichiers de type **.BAS**

La commande REN ou RENAME

La commande **REN** vous permet de changer le nom (**REN**ommer) d'un fichier existant. Le nouveau nom est spécifié en premier suivi du signe égal (=), puis du nom du fichier à renommer. Si le nouveau nom existe déjà sur la disquette, un message d'erreur s'affiche.

Les noms de fichiers comportant des jokers ne rentrent pas dans le cadre de cette commande intégrée ; il faut donc faire appel au programme transitoire **RENAME.COM**.

REN NOUVNOM.BAS=ANCNOM.BAS donne au fichier **ANCNOM.BAS** le nouveau nom **NOUVNOM.BAS**

REN B:NOUVNOM.BAS=ANCNOM.BAS donne au fichier **ANCNOM.BAS** le nouveau nom **NOUVNOM.BAS**, sur B.

La commande TYPE ou TYP

La commande **TYPE** (imprimer) demande que le fichier spécifié défile à l'écran. Si ce fichier n'est pas de type ASCII, des choses imprévisibles risquent d'apparaître à l'écran.

```
TYPE KEYS.CCP
```

...affiche le fichier **KEYS.CCP**.

La commande USER, ou USE

USER change le numéro d'utilisateur courant. CP/M Plus s'initialise avec le numéro d'utilisateur courant 0. Vous ne pouvez accéder qu'aux fichiers identifiés par le numéro d'utilisateur courant, ce qui permet de partager le catalogue.

Il est possible d'accéder à un fichier de l'utilisateur 0 ayant « **SYS** » comme attribut à partir de tous les autres numéros d'utilisateur. Cette puissante fonction permet de rendre les utilitaires et les programmes d'applications accessibles à tous les utilisateurs sans avoir à les copier dans chaque zone utilisateur.

```
USER 3
```

définit l'utilisateur courant numéro 3.

Les commandes transitoires

Pour réaliser sur vos fichiers des opérations un peu plus sophistiquées qu'avec le mode direct, vous pouvez faire appel à l'un des nombreux programmes utilitaires fournis avec le CP/M. Vous les appelez simplement en tapant leur nom, suivi parfois d'un nom de fichier et/ou d'un ou plusieurs paramètres. Vous avez déjà probablement utilisé **DISCKIT3**.

Nous avons regroupé ces commandes par catégories. Vous en trouverez une documentation complète dans le Guide du CP/M Plus.

Les commandes **DISCKIT3**, **SETKEYS**, **SETLST**, **SETSIO**, **PALETTE**, **LANGUAGE** et **AMSDOS**, ainsi que les programmes de gestion d'écran GSX et l'installation de **LOGO3**, propres à Schneider , ne fonctionnent que sur son système et ne peuvent être employées sur les autres systèmes CP/M.

Vous pouvez entrer plusieurs commandes sur une même ligne en séparant les commandes par un point d'exclamation. Exemple :

```
LANGUAGE 3!SETKEYS KEYS.WP
```

La gestion des périphériques

DISCKIT3 formate, copie et vérifie les disquettes. Il est plus rapide de formater pendant la copie que d'effectuer les deux opérations en deux temps. Des menus vous indiquent les touches à activer (essentiellement des touches de fonction). Les disquettes commercialisées sont similaires aux disquettes système sinon qu'elles sont adaptées à la distribution de logiciels ; toutefois, les disquettes formatées pour le stockage des données sont peut-être plus adéquates pour travailler sous CP/M Plus.

AVERTISSEMENT

Le brevet de votre disquette CP/M (qui possède un numéro de série codé électroniquement) vous permet de l'utiliser sur un seul micro-ordinateur. Ceci signifie en particulier qu'il vous est interdit de faire don d'une copie de votre CP/M (copie qui posséderait alors votre numéro de série). Etant donné que chacune des copies de la face 1 des disquettes système contient votre CP/M (dans le fichier **.EMS**), il vous est interdit de vendre ou échanger toute disquette contenant ce fichier ou de vous en séparer de quelque façon que ce soit.

Jeux de caractères

Le CPC6128 possède un jeu complet de caractères internationaux. La commande **LANGUAGE** modifie certains caractères pour permettre à des logiciels peu perfectionnés d'afficher à l'écran d'autres caractères et des lettres accentuées. Pour plus de détails, consultez la partie 16 du chapitre « Pour information ».

La commande :

```
LANGUAGE 3
```

permet d'installer le jeu de caractères britannique, remplaçant le signe # par £ (le jeu américain est le jeu par défaut).

Les couleurs

Sous CP/M Plus, les couleurs par défaut du 6128 (avec moniteur couleur) affichent des caractères en blanc brillant sur fond bleu. Vous pouvez les modifier par la commande **PALETTE**, qui possède plusieurs paramètres, un pour chaque encre : l'encre 0 inclut le fond et la bordure de l'écran et l'encre 1 définit la couleur du texte. Chaque couleur est codée par un nombre compris entre 0 et 63. Sur un moniteur monochrome, ce numéro définit l'intensité d'affichage.

Vous pouvez spécifier n'importe quel numéro d'encre compris entre 1 et 16, mais seules les deux premières encres sont visibles en mode 80 colonnes.

La commande :

```
PALETTE 63,1
```

inverse les spécifications normales des encres 0 et 1, changeant le fond en blanc brillant (63) et le texte en bleu (1).

Pour sélectionner les couleurs (ou leurs intensités), aidez-vous du tableau de la page suivante. Vous pouvez au choix utiliser la représentation décimale ou hexadécimale.

Couleur	Code hexadécimal	Code décimal	Couleur	Code hexadécimal	Code décimal
Noir	&00	0	Bleu pastel	&2B	43
Bleu	&02	2	Orange	&2C	44
Bleu vif	&03	3	Rose	&2E	46
Rouge	&08	8	Magenta pastel	&2F	47
Magenta	&0A	10	Vert vif	&30	48
Mauve	&0B	11	Vert marin	&32	50
Rouge vif	&0C	12	Turquoise vif	&33	51
Violet	&0E	14	Vert citron	&38	56
Magenta vif	&0F	15	Vert pastel	&3A	58
Vert	&20	32	Turquoise pastel	&3B	59
Turquoise	&22	34	Jaune vif	&3C	60
Bleu ciel	&23	35	Jaune pastel	&3E	62
Jaune	&28	40	Blanc brillant	&3F	63
Blanc	&2A	42			

Le clavier

Vous pouvez modifier les codes générés par le clavier à l'aide de la commande **SETKEYS**. Ceci permet d'affecter les codes adéquats aux touches physiques ou logiques. Les codes doivent être inscrits dans un fichier dont le nom est ensuite passé en paramètre dans la commande **SETKEYS**. Ce fichier de commande peut être créé par un éditeur de texte, **PIP**, ou même à partir du BASIC. Par exemple :

```
SETKEYS KEYS.TST
```

où le fichier **KEYS.TST** contient :

E &8C « **DIR ↑M** » touche logique 12
8 N S C « **↑H** » espace arrière = **[CONTROL]H**, 08 ASCII

Ceci modifie la touche logique **[CONTROL] [ENTER]** (code &8C), qui devient **DIR [RETURN]**, et transforme la touche de retour arrière du curseur ⇦ (touche numérique 8) en espace arrière.

Les fichiers standard du 6128 sont **KEYS.CCP** pour les commandes CP/M, **KEYS.DRL** pour le Dr. LOGO (face 3 de la disquette système) et **KEYS.WP** qui convient à de nombreux traitements de texte.

Les imprimantes

Les imprimantes peuvent être initialisées par la commande :

SETLST <nomfich>

le « nomfich » contient la ou les chaînes à envoyer à l'imprimante. Comme avec le fichier de commande pour SETKEYS, les codes de contrôle peuvent être représentés par :

↑ <caractère>

ou

↑'<valeur du caractère>'

ou bien

↑'<nom du code de contrôle>'

Les noms des codes de contrôle sont ESC, FF, etc., comme l'indique le tableau des caractères ASCII du chapitre « Pour information ».

Pour de nombreuses imprimantes, la valeur 15 représente un code d'initialisation utile, définissant une impression condensée.

La commande :

```
PRINT #8,CHR$(15)
```

serait la syntaxe en BASIC. Sous CP/M, elle devient :

```
SETLST CONDENSE
```

Le fichier **CONDENSE** contient l'une des lignes ci-dessous :

```
↑'SI'
↑O
↑'&F'
↑'15'
```

qui représentent toutes la valeur décimale 15.

Pour certains programmes d'application, l'écran doit être de 24 lignes x 80 colonnes. La commande **SET24X80** permet d'obtenir ce format d'écran.

Pour activer le mode 24x80, utilisez les commandes :

```
SET24X80
```

ou

```
SET24X80 ON
```

Pour le désactiver, utilisez la commande :

```
SET24X80 OFF
```

La taille d'écran normale du CPC6128 est 24 x 80, la dernière ligne étant réservée aux messages d'état. La désactivation du mode 24 x 80 n'est visible que si la ligne d'état est également invalidée. Pour la validation et l'invalidation de la ligne d'état, reportez-vous à la partie 15 du chapitre 7.

Interface série

CP/M Plus permet de gérer une interface RS232 ne possédant qu'un seul canal. Pour visualiser ses principales caractéristiques, tapez la commande **SETSIO** (sans paramètre) :

```
SETSIO
```

ou une commande comprenant l'une (voire la totalité) de ces sélections :

```
SETSIO, RX 1200, TX 75, PARITY NONE, STOP 1, BITS 8,
HANDSHAKE ON, XOFF OFF
```

pour une nouvelle configuration.

Le débit de l'interface et l'état **XON/XOFF** peuvent également être fixés par la commande **DEVICE** qui gère les périphériques logiques et physiques. Les périphériques logiques sont indiqués par un deux points (:). Pour visualiser tous les attributs des périphériques installés, tapez :

```
DEVICE
```

Pour les modifier : un périphérique.

`DEVICE SIO[1200]`	fixe le débit à 1200 bauds.
`DEVICE SIO[XON]`	active le protocole **XON/XOFF**.
`DEVICE SIO[NOXON]`	désactive le protocole **XON/XOFF**.]

Vous pouvez également modifier les affectations entre périphériques logiques et physiques. Habituellement, **CON:** correspond à **CRT** (clavier/écran), **AUX:** à **SIO** (interface série en option) et **LST:** à **LPT** (interface Centronics pour imprimante). La commande :

```
DEVICE LST:=SIO
```

envoie les données destinées à l'imprimante vers l'interface série (si celle-ci est installée).

Il s'agit d'un réacheminement de canal, à ne pas confondre avec la certaines commandes **PIP**. Les deux commandes **GET** <nom de fichier> et **PUT** <nom de fichier> réacheminent les données à destination et en provenance de la console et les sorties vers l'imprimante en les dirigeant vers un fichier particulier et non pas vers un périphérique.

```
PIP
```

L'utilitaire **PIP** (Programme d'Interconnexion aux Périphériques) vous permet d'échanger des informations entre l'ordinateur et ses périphériques. La forme de ces commandes est généralement :

PIP <destination> = <source>

La <source> et la <destination> peuvent être soit un nom de fichier (jokers acceptés) pour la source, soit le code du périphérique. On peut utiliser les codes suivants :

En tant que source	**En tant que destination**
CON : entrée console	**CON** : sortie console
AUX : entrée auxiliaire	**AUX** : sortie auxiliaire
EOF : manque de fin de fichier	**LST** : imprimante
	PRN : imprimante avec tabulations supplémentaires, numérotation de lignes et sauts de page.

Exemples :

```
PIP B:=A:*.COM
```

...copie tous les fichiers avec extension **.COM** de l'unité A sur l'unité B.

```
PIP KEYBOARD.CPM=KEYS.CCP
```

...réalise une copie de **KEYS.CCP** et l'appelle **KEYBOARD.CPM**.

```
PIP CON:=KEYS.CCP
```

...envoie le fichier **KEYS.CCP** à l'écran (même effet que **TYPE KEYS.CCP**).

```
PIP LST:=KEYS.CCP
```

...envoie le fichier **KEYS.CCP** sur l'imprimante.

```
PIP TYPEIN.TXT=CON:
```

...place des données venant du clavier dans le fichier **TYPEIN.TXT**.

L'opération s'achève par un **[CONTROL]Z**. Pour accéder à une nouvelle ligne, vous devez envoyer à chaque fois un **[CONTROL]J** (code ASCII du retour à la ligne) après **[RETURN]**.

Si vous tapez **PIP** sans paramètre, vous recevez le message * : vous pouvez alors entrer les commandes qui vous intéressent. Ce procédé est particulièrement utile pour la copie de fichiers lorsque le fichier **PIP.COM** ne se trouve ni sur la disquette source, ni sur la disquette cible. Vous pouvez charger **PIP** à partir de la face 1 d'une des disquettes système, retirer cette disquette et insérer les disquettes requises pour la copie.

Pour sortir de **PIP**, appuyez sur **[RETURN]** à l'apparition du message *.

Notez que **PIP** ne peut effectuer de copies de fichiers sur un système à une seule unité. Utilisez alors **FILECOPY**.

Gestion du système

Les programmes transitoires **DIR**, **ERASE**, **RENAME** et **TYPE** offrent plus de possibilités que leurs homologues intégrés. Comme de nombreux programmes transitoires de Digital Research, leurs paramètres secondaires sont indiqués entre crochets. Tous les détails les concernant figurent dans les fichiers **HELP** (face 3 d'une des disquettes système). En voici quelques exemples :

`DIR [FULL]`	affiche la taille et les attributs des fichiers
`ERASE *.COM [CONFIRM]`	vous demande confirmation pour chaque fichier rencontré

`RENAME`	vous demande de taper l'ancien et le nouveau nom de fichier
`RENAME *.SAV=*.BAK`	renomme tous les fichiers à suffixe **.BAK** en fichiers **.SAV**
`TYPE KEYS.WP [NOPAGE]`	supprime la pagination de l'écran

Nous avons déjà évoqué les attributs de fichiers **SYS** (système) et **RO** (lecture seulement). La commande **SET** qui accepte les jokers, permet de modifier ces attributs et de nombreux autres.

Les commandes :

```
SET *.COM [RO]
SET KEYS.CCP [RO]
SET A: [RO]
```

affectent l'attribut **RO** (lecture seulement) aux fichiers spécifiés ou au disque tout entier, de façon à empêcher un effacement accidentel.

Les commandes :

```
SET *.COM [RW]
SET KEYS.CCP [RW]
SET A: [RW]
```

affectent l'attribut **RW** (lecture/écriture) à ces mêmes fichiers.

Les commandes :

```
SET *.COM [SYS]
SET KEYS.CCP [SYS]
```

affectent l'attribut **SYS** aux fichiers spécifiés. Les fichiers affectés de cet attribut ne seront pas listés par la commande **DIR** (utilisez alors **DIRS**, ou **DIRSYS**). Cependant, ils peuvent être utilisés et, de plus, s'ils se trouvent dans la zone utilisateur 0, ils seront disponibles dans toutes les autres zones utilisateur.

Les commandes :

```
SET *.COM [DIR]
SET KEYS.CCP [DIR]
```

retirent l'attribut **SYS**.

Vous pouvez affecter aux disquettes une étiquette, un nom ou un mot de passe réservant l'accès de l'ensemble du catalogue plutôt que celui des fichiers qu'il contient. Vous pouvez cependant attribuer des mots de passe à des fichiers, individuellement.

```
SET [NAME=ROLAND]
SET [PASSWORD=LUCIEN]
SET [PROTECT=ON]
```

agissent sur la disquette de l'unité par défaut.

```
SET *.*[PASSWORD=LUCIEN]
SET *.*[PROTECT=READ]
```

agissent sur les fichiers de la disquette de l'unité par défaut (les jokers *.* indiquent ici que tous les fichiers de la disquette sont concernés par la commande).

La commande **INITDIR** (face 2 d'une des disquettes système) permet de « marquer » une disquette, en vue de la prise en compte de la date et de l'heure lors d'une création ou mise à jour de fichier. Les commandes :

```
INITDIR
SET [CREATE=ON]    ... ou...    [ACCESS=ON]  ... et...
SET [UPDATE=ON]    ...avec...
DIR [FULL]
```

permettent de « marquer » la disquette (**INITDIR**), puis de spécifier que la date et l'heure seront prises en compte pour la création et la mise à jour de fichier. La commande **DIR [FULL]** permet de visualiser le catalogue de la disquette avec affichage de la date et de l'heure de création et de mise à jour. Vous devez alors taper :

```
DATE SET
```

à chaque lancement de CP/M Plus pour mettre l'horloge du système à l'heure. Elle sera ensuite mise à jour régulièrment par votre ordinateur et vous pourrez la vérifier par les commandes :

```
DATE
```

...et...

```
DATE CONTINUOUS
```

AVERTISSEMENT

Si vous attribuez à une disquette des mots de passe, une étiquette ou l'indication de la date et du jour, il est recommandé de ne plus y écrire de données sous AMSDOS ou CP/M 2.2 car aucun des deux ne possèdent ces fonctions.

Sauf spécification contraire, vous ne pourrez accéder qu'aux fichiers de l'unité par défaut. La commande :

```
SETDEF *,A:
```

(où * renvoie à l'unité par défaut) ordonne au CP/M (lors de la recherche de fichiers) de procéder d'abord dans l'unité par défaut, puis dans l'unité **A**. Autrement dit, si **B** est l'unité par défaut, les fichiers seront automatiquement retrouvés même s'ils n'existent que dans l'unité **A**.

Les commandes :

```
SETDEF [PAGE]
SETDEF [NOPAGE]
```

activent et désactivent la pagination automatique de la console.

La plupart des paramètres spécifiés dans les commandes **DEVICES**, **SET** et **SETDEF** doivent être initialisés, tout comme la date, à chaque lancement de CP/M Plus, surtout lorsqu'ils se rapportent aux unités de disquettes (plutôt qu'à des fichiers ou des disquettes). Ceci est essentiel pour obtenir un fichier **PROFILE.SUB** correct.

SUBMIT est nécessaire à l'exécution automatique de fichiers de commandes. Ces fichiers de commandes contiennent du texte et il est possible d'y ajouter des lignes de données pour des programmes si ces dernières commencent par le caractère <.

Différentes versions de la commande **SHOW** permettent d'afficher la taille de l'unité, l'espace disponible, le nombre d'entrées du catalogue disponibles sur la disquette avec les zones utilisateur contenant les fichiers et, s'il existe, le label de la disquette :

```
SHOW B:
SHOW B:[LABEL]
SHOW B:[USERS]
SHOW B:[DIR]
SHOW B:[DRIVE]
```

toutes ces commandes se rapportent à l'unité **B**.

Abandon de CP/M Plus

```
AMSDOS
```

Ce programme permet l'abandon du contrôle par CP/M pour revenir au BASIC Schneider , d'où les commandes AMSDOS de gestion des disquettes sont disponibles.

Programmation avancée

Les programmes stockés sur la face 2 d'une des disquettes système serviront essentiellement aux spécialistes. Nous vous recommandons de consulter le guide du CP/M Plus, SOFT971 ou d'autres ouvrages de référence.

Exploitation sous CP/M 2.2

Cette partie insiste sur les différences de fonctionnement du système exploité sous CP/M 2.2.

CP/M 2.2 est chargé à partir des deux premières pistes d'une des disquettes système. Le secteur d'amorçage diffère de celui du CP/M Plus ; veillez à ne pas les confondre. Théoriquement, vous pouvez utiliser, indifféremment dans l'une ou l'autre des unités, des disquettes commercialisées, IBM ou formatées pour des données, mais, pour des raisons de fonctionnement du système, vous devrez généralement limiter leur emploi à la deuxième unité.

En dehors de l'intervention d'un utilitaire CP/M (tel que **FILECOPY**), CP/M 2.2 ne vous permet pas d'écrire sur une disquette sans qu'elle ait préalablement été amorcée. De plus, le type de formatage (Système, Data, IBM) n'est pris en compte que lors de l'amorçage. Pour l'unité principale (A), cet amorçage se produit chaque fois que CP/M 2.2 repasse en mode direct ou que l'on envoie un **[CONTROL]C** après le message d'interrogation **A**> ou **B**>. L'unité supplémentaire (B) est amorcée la première fois que l'on y accède, l'unité A étant amorcée.

Si vous essayez d'écrire sur une disquette qui n'a pas été amorcée, le message d'erreur :

Bdos Err on <drive> : **R/O**

...apparaît. Appuyez sur n'importe quelle touche pour continuer. Si, de surcroît, la disquette est dotée d'un format différent, une erreur en lecture ou écriture se produit. Tapez **C** pour poursuivre.

Si vous achetez un logiciel sur disquette, vous devez le copier sur une disquette système CP/M 2.2 à l'aide de **FILECOPY** ou **PIP**, ou convertir la disquette en disquette système en y ajoutant votre CP/M, à l'aide des commandes **BOOTGEN** et **SYSGEN**. Lisez attentivement le contrat d'utilisation des logiciels de l'annexe 1.

SYSGEN, sans paramètre, est le programme de copie par excellence qui vous indique par des messages l'insertion des disquettes source et cible, et copie les pistes CP/M 2.2 d'une disquette sur l'autre. De même, **BOOTGEN** copie le secteur 1 de la piste 0 (le chargeur) et le secteur de configuration.

La commande **DIR** n'accepte pas de paramètre hormis un identificateur de fichier. Les fichiers apparaissent selon leur ordre d'entrée sur le répertoire de la disquette.

La commande **STAT** possède certaines fonctions de base de SET et SHOW. Les commandes :

```
STAT
STAT A:
STAT B:
```

...affichent l'état de la disquette et l'espace disponible.

```
STAT *.COM
STAT EX1.BAS
```

...affichent des compléments d'information sur un fichier particulier.

```
STAT *.COM $R/O
STAT EX1.BAS $R/O
```

...attribuent à un fichier le statut 'lecture seulement', de façon à ce qu'il ne puisse pas être détruit accidentellement.

```
STAT *.COM $R/W
STAT EX1.BAS $R/W
```

...remplacent un fichier au statut normal de lecture/écriture.

```
STAT *.COM $SYS
STAT SECRET.BAS $SYS
```

...attribuent à un fichier un statut « système » qui le rend invisible sur le catalogue de la disquette et inaccessible aux programmes de copie de fichiers tout en restant disponible à tout autre usage.

```
STAT *.COM $DIR
STAT SECRET.BAS $DIR
```

...inversent le statut précédent. Redonnent au fichier son statut visible.

L'utilitaire **FILECOPY** vous permet de copier des fichiers d'une disquette sur une autre avec un système à une seule unité. Il vous donne les instructions nécessaires pour les changements de disquettes. Si vous utilisez des noms de fichiers avec jockers, il vous demandera confirmation entre chaque fichier puis affichera le nom des fichiers au fur et à mesure de la copie.

```
FILECOPY *.COM
```
...copie tous les fichiers **.COM**.

```
FILECOPY PIP.COM
```
...copie le fichier **PIP.COM**.

DISCKIT2 assure les mêmes fonctions que **DISCKIT3** mais, disposant de moins de mémoire, il copie les disquettes plus lentement.

Deux utilitaires sont disponibles pour des transferts entre cassette et disquette. Seuls les spécialistes trouveront toutefois un intérêt au transfert de fichiers autres que des textes en codes ASCII.

CLOAD (cassette LOAD) accepte deux paramètres : le nom du fichier source (sur cassette) entre guillemets, et le nom du fichier destination (sur disquette). Si ce dernier est omis, le fichier sur disquette porte le même nom que celui sur cassette (obligatoirement compatible avec le CP/M). Si le nom du fichier source est omis, **CLOAD** lit le premier nom qu'il rencontre sur la bande. Si le premier caractère du nom du fichier cassette est un!, les messages liés à l'utilisation des cassettes sont alors supprimés.

Exemple :

```
CLOAD "MA LETTRE" MALETTRE.TXT
```

CSAVE (cassette SAVE) accepte trois paramètres, les deux premiers étant le nom du fichier source (sur disquette) et le nom du fichier destination (sur cassette), entre guillemets. Si le nom du fichier destination est omis, ce nom sera celui de la source. Si le premier caractère du nom du fichier cassette est un !, les messages 'cassette' sont supprimés. Si les deux noms sont spécifiés, on peut alors utiliser un troisième paramètre spécifiant la vitesse d'enregistrement : **Ø** pour 1000 bauds, **1** pour 2000 bauds.

Exemples :

```
CSAVE MALETTRE.TXT "MA LETTRE TEXTE" 1
CSAVE FICHIER
```

SETUP

Cet utilitaire vous permet de redéfinir les caractéristiques du clavier de votre 6128, du lecteur de disquette et de l'interface série ainsi que de lancer diverses actions au moment du chargement du CP/M 2.2

Contrairement aux utilitaires de CP/M Plus qui remplissent leur fonction après leur exécution, **SETUP** modifie le secteur de configuration, appelé lors de l'amorçage à froid de la disquette. Il rappelle à cet égard l'action de l'utilitaire **PROFILE.SUB**.

Ce programme est dirigé par un menu : lorsqu'un écran est correct ou ne requiert aucune modification, vous passez au suivant en répondant **Y** à la question :

```
Is this correct (Y/N):_
```

Il est possible d'arrêter le programme à tout moment par la commande **[CONTROL]:C**. Une fois tous les chargements effectués, il vous demande :

```
Do you want to update your system disc (Y/N):_
```

(Confirmez-vous cette remise à jour de la configuration de votre système)

...qui vous permet de revenir à l'ancienne configuration (en tapant **N**).

```
Do you want to restart CP/M (Y/N):_
```

...permettant d'initialiser et donc de tester votre nouvelle configuration en répondant **Y**.

Pour copier la zone où sont sauvegardées les variables du système, d'une disquette sur une autre, vous pouvez soit utiliser **BOOTGEN** (voir plus loin), soit charger **SETUP** du disque source, en répondant **Y** à la question **« Do you want to update your system disc (Y/N) : - »**

Les caractères qui ont une valeur ASCII inférieure à 32 peuvent être entrés à l'intérieur d'une chaîne alphanumérique en tapant suivi du caractère ↑ convenable tiré de la série @, A...Z, [,\,], >, -.

Les options suivantes reviennent le plus souvent :

**** Initial command buffer (zones de commandes à l'initialisation) :**

Tous les caractères entrés ici apparaîtront comme s'ils avaient été tapés en mode direct au moment du chargement du CP/M, permettant ainsi de lancer automatiquement un programme. Souvenez-vous d'y inclure l'équivalent de la touche **[RETURN]**, représentée par les deux caractères ↑ **M**.

Ainsi, pour le lancement automatique de la commande **DIR**, vous devez taper dans cette zone :

```
DIR↑M
```

Pour le lancement automatique du programme Dr. LOGO, tapez dans cette zone :

```
SUBMIT LOGO2↑M
```

Sign-on string (le message d'introduction) :

C'est le message affiché en haut de l'écran quand on charge le CP/M. Notez l'utilisation de ↑ **J** ↑ **M** pour l'entrée du saut de ligne et du retour chariot. La première partie de ce message définit la couleur de l'écran et celle de la bordure pour le mode 80 colonnes (elle doit être recopiée telle quelle si les caractéristiques restent inchangées).

Keyboard translations (changement des codes clavier) :

Donne de nouvelles valeurs ASCII aux touches du clavier, comme le fait la commande **'KEY DEF'** du BASIC. Les paramètres requis sont les codes clés et les valeurs ASCII à leur attribuer. Vous trouverez une illustration des codes clés dans le diagramme figurant en haut à droite de l'ordinateur ou à la partie 4 du chapitre « Pour information ».

Keyboard expansions (extensions du clavier)

Emule la commande BASIC **'KEY'**.

Pour terminer...

Les commandes **DISCKIT2**, **SYSGEN**, **BOOTGEN**, **FILECOPY**, **SETUP**, **CSAVE** et **CLOAD** sont conçues pour les systèmes AMSTRAD et ne fonctionnent que sous CP/M 2.2. Elles n'ont aucune fonction sous les autres systèmes CP/M mais certains fabricants proposent des utilitaires identiques (portant souvent le même nom) adaptés à leurs systèmes.

Les programmes CP/M 2.2 ci-dessous, destinés aux spécialistes, sont également disponibles sur la face 4 d'une des disquettes système. L'utilisateur peut à leur sujet consulter le guide du CP/M, SOFT159, ainsi que d'autres ouvrages de référence.

ASM	Assembleur du 8080
DDT	Utilitaire de mise au point des codes d'assemblage du 8080
DUMP	Utilitaire de vidage de fichier sous forme hexadécimale
ED	Editeur de texte simple
LOAD	Convertit les fichiers **. HEX**, créés avec **ASM**, en fichiers **. COM**
MOVCPM	Crée le CP/M 2.2 avec une taille TPA réduite
SUBMIT	Utilitaire d'exécution de fichier de commandes
XSUB	Utilitaire de traitement par lots

Ces programmes sont réservés au CP/M 2.2 (face 4) et ne doivent pas être confondus avec d'autres programmes stockés sur les faces 1, 2 et 3 des disquettes système pour exploitation sous CP/M Plus.

Chapitre 6
Introduction au LOGO

L'objectif de ce chapitre est de vous faire prendre contact avec LOGO. Assorti d'exemples, il est complété d'un guide des commandes disponibles mais n'est ni un cours exhaustif, ni un guide de référence. Il existe pour cela toute une série de publications AMSOFT et d'autres ouvrages.

Sujets abordés :

* Le concept de LOGO
* Chargement et mise en service de Dr. LOGO
* Le graphisme de la tortue
* Ecrire vos propres procédures
* Editer vos propres procédures

Qu'est-ce que le LOGO ?

Le LOGO peut vous aider à devenir un programmeur, que vous ayez ou non des notions d'informatique. C'est un langage puissant qui devient de plus en plus populaire en raison de sa simplicité.

Vous vous servez de « procédures » pour construire les différentes parties de votre programme. Le Dr. LOGO a sa propre collection de « procédures » appelées « primitives » que vous utiliserez également pour programmer.

Dans les années 70, une équipe de chercheurs en informatique et d'éducateurs dirigés par Seymour Papert mit au point le LOGO et sa petite tortue graphique pour permettre à de très jeunes enfants de programmer un ordinateur. La tortue est pour ces jeunes apprentis, comme le dit Papert, un « objet pour penser », un outil pour apprendre d'une façon différente. En forme de tête de flèche, elle peut être déplacée sur l'écran à l'aide de commandes simples.

Dr. LOGO

Le Dr. LOGO est une version très élaborée du LOGO. Elle a été spécialement adapté à l'ordinateur individuel Schneider afin de le rendre encore plus simple à programmer. Des extensions ont permis d'y inclure la possibilité d'utiliser les puissantes facultés sonores du CPC6128 tandis que la correction des programmes est facilitée par le pavé de gestion du curseur.

Préparons-nous

Pour mettre en service le Dr. LOGO, insérez une copie de la face 1 d'une des disquettes système dans l'unité de disquette et tapez :

```
|cpm
```

Lorsque le message **A** > apparaît, retirez la disquette et insérez une copie de la face 3 (DR. LOGO & HELP).

Pour lancer le Dr. LOGO, tapez :

```
SUBMIT LOGO3
```

Au bout de quelques secondes, un message de bienvenue s'affiche, suivi d'un point d'interrogation.

Dr. LOGO sous CP/M 2.2

Remarque

La version du Dr. LOGO stockée sur la face 4 d'une des disquettes système a été prévue pour une utilisation du Dr. LOGO sous CP/M 2.2 (sur Schneider CPC664 ou CPC464+DDI1). Elle n'est donc pas recommandée sur le CPC6128. La face 3 contient toutes les spécifications du Dr. LOGO de Digital Research.

Si vous préférez malgré tout utiliser la version CP/M 2.2 du Dr. LOGO, insérez dans l'unité une copie de la face 4 d'une des disquettes système et tapez :

```
|CPM
```

Lorsque **A** > apparaît, tapez :

```
SUBMIT LOGO2
```

Au bout de quelques secondes s'affiche un message de bienvenue, suivi d'un point d'interrogation.

Les premiers pas

Le point d'interrogation vous indique que Dr. LOGO est prêt à recevoir vos instructions au clavier.

Essayez de taper en minuscules :

```
fd 60 [RETURN]
```

... et vous verrez apparaître une tortue (une flèche grand format) qui avancera de 60 unités en laissant un trait derrière elle. L'écran va se restructurer en ménageant une zone importante pour le graphique et un emplacement plus petit pour le texte (?) en bas de l'écran.

Dr. LOGO décide souvent de changer l'organisation de l'écran avec soit une grande zone pour le texte, soit une grande zone pour le graphisme, selon votre besoin.

Tapez :

```
rt 90 [RETURN]
```

...et la tortue bouge de 90 degrés vers la droite.

Maintenant tapez :

```
fd 60
```

...et une autre ligne se dessine de la même longueur que la première et à angle droit.

Entraînez-vous avec les instructions **fd**, **bk** (back : arrière) **rt** et **lt** (left : gauche), pour voir ce qui se passe sur l'écran.

Les « procédures » de Dr. LOGO

Une procédure est une liste d'instructions accomplissant un certain travail. Vous allez sûrement écrire vos premières procédures en utilisant les instructions de Dr. LOGO appelées « primitives ».

fd, **bk**, **rt** et **lt** sont des primitives de Dr. LOGO utilisables à tout moment pour construire vos propres « procédures ». Une autre primitive très utilisée est **cs**, qui nettoie l'écran et renvoie la tortue à la case de départ.

L'élaboration d'une procédure simple

Vous pouvez également constater que si la formule :

```
fd 60 rt 90
```

...est répétée 4 fois, elle dessine un carré de 60 unités de côté.

Vous pouvez obtenir le même résultat en écrivant très simplement :

```
repeat 4 [fd 60 rt 90]
```

Videz l'écran et tapez la formule pour voir ce qui se passe.

Pour faire de cette formule une nouvelle procédure, appelée « **carre** », tapez :

```
to carre
repeat 4 [fd 60 rt 90]
end
```

Dr. LOGO comprend maintenant « **carre** » et, chaque fois qu'il le rencontrera, il tracera un carré sur l'écran. Nous aurions pu donner n'importe quel nom à notre procédure, mais « **carre** » est un bon moyen pour s'en souvenir.

Dr. LOGO nous permet de taper toutes sortes de combinaisons du style : **carre rt 45 carre**, qui va dessiner 2 carrés, le second ayant un angle de 45 degrés par rapport au premier.

Procédures avec paramètres

Il est possible de bâtir des procédures auxquelles nous pourrons demander « combien », de même que pour les primitives. Pour bâtir une procédure qui va construire des carrés de n'importe quelle taille, nous allons redéfinir « **carre** » :

```
to carrequelconque :cote
repeat 4 [fd :cote rt 90]
end
```

Vous remarquerez que la variable :cote est précédée de 2 points qui indiquent à Dr. LOGO que **:cote** est une variable et non pas une commande.

Lorsque nous utiliserons la procédure **carrequelconque**, **:cote** doit recevoir une valeur. Ainsi, **carrequelconque 150** donnera un carré de 150 unités de côté.

Essayons de combiner 2 procédures pour voir ce qu'il advient. Par exemple, à partir d'une instruction :

```
cs carrequelconque 100 rt 45 carrequelconque 150
```

...la tortue va dessiner 2 carrés de côtés différents, l'un faisant un angle de 45 par rapport à l'autre.

Notez comment Dr. LOGO utilise un point d'exclamation ! pour vous rappeler qu'une ligne de commandes s'est divisée en plusieurs lignes sur l'écran.

Utilisation de variables pour stocker des valeurs

Dr. LOGO vous permet d'utiliser des variables pour stocker des valeurs et en transmettre à une procédure.

Elaborez tout d'abord une procédure appelée triangle :

```
to triangle
repeat 3 [fd :bord rt 120]
end
```

Vous pouvez tester ceci en tapant :

```
make "bord 100
triangle
```

Si nous voulons connaître la valeur du bord, nous entrons **:bord** après le ? et Dr. LOGO nous affiche sa valeur.

Enfin, nous pouvons utiliser notre variable **:bord** dans une nouvelle procédure afin de construire un dessin. Remarquez comment la valeur de **:bord** est augmentée de façon à agrandir notre dessin à chaque passage.

```
to dessin
triangle lt 60 triangle rt 60
make "bord :bord+4
dessin
end
make "bord 10
cs dessin
```

Quand vous en avez assez, appuyez sur **[ESC]** pour arrêter le programme.

La correction des programmes et des procédures

Dr. LOGO vous permet de corriger les fautes de frappe et de modifier les procédures que vous avez définies. Les touches utilisées pour la correction sont :

Les touches curseur ⇧⇩⇦⇨ pour le déplacement du curseur d'un caractère ou d'une ligne à la fois.

Les touches curseur ⇧⇩⇨⇦ combinées à la touche **[CONTROL]** pour se déplacer d'une extrémité de l'écran à l'autre.

[CLR] efface le caractère sous le curseur. **[DEL]** efface le caractère à gauche du curseur.

[RETURN] indique au Dr. LOGO que la correction d'une ligne de commande est terminée ou bien donne la ligne suivante lors de la correction d'une procédure.

[ESC] signifie fin de correction et **[COPY]** signale au Dr. LOGO que la correction de la procédure est terminée.

Quand vous entrez des commandes ou de nouvelles procédures, corrigez le texte en face de vous sur l'écran. Tout autre caractère que ceux mentionnés plus haut est inséré dans le texte à la position du curseur.

Pour la correction d'une procédure existante, utilisez la commande **ed**

Dr. LOGO affiche alors l'ancienne version de la procédure sur l'écran et vous permet d'effectuer ainsi vos modifications à l'aide des commandes de gestion du curseur.

Essayez la correction de la procédure **dessin** en tapant **ed"dessin**

Servez-vous des touches de correction. Quand vous avez terminé, appuyez sur la touche **[ESC]**. Dr.LOGO abandonne alors le mode correction et vous redonne la version originale inchangée de votre procédure.

Tapez **ed"dessin** à nouveau puis changez le 4 en 8, appuyez sur la touche **[COPY]** pour sortir et lancez la procédure pour visualiser le graphique résultant. N'oubliez pas de donner une valeur à **:bord**.

Quelques mots sur le fonctionnement

L'espace de travail utilisé par Dr. LOGO est divisé en nœuds. Vous pouvez savoir combien il vous en reste en tapant :

```
nodes
```

Parfois, lorsque presque tous les nœuds sont utilisés, Dr. LOGO fera le ménage de l'espace de travail et la tortue fera une pause. On peut aussi demander à Dr. LOGO de faire ce nettoyage par la commande :

```
recycle
```

Ceci pour vous permettre de continuer après que Dr. LOGO se soit plaint de n'avoir plus de nœuds disponibles.

Si vous utilisez la version CP/M 2.2 du Dr. LOGO (face 4), assurez-vous, avant de lancer Dr. LOGO, qu'il y a suffisamment de place sur votre disquette pour vous permettre de sauvegarder vos procédures. Utilisez la commande **CAT** d'AMSDOS (voir la partie 7 du Cours Élémentaire).

Jetez un coup d'œil sur les paragraphes suivants et essayez quelques exemples. Vous ne comprendrez pas tout la première fois, mais plus vous avancerez dans votre apprentissage de Dr. LOGO, plus vous utiliserez de commandes différentes.

Quand vous avez terminé avec Dr. LOGO, tapez :

```
bye
```

Liste des primitives de Dr.LOGO

Ce paragraphe dresse la liste alphabétique des primitives du Dr.LOGO par groupes, en donnant la forme syntaxique, parfois accompagnée d'un exemple.

Les commandes accompagnées d'un astérisque ne sont pas disponibles avec la version CP/M 2.2 du Dr. LOGO (face 4 d'une des disquettes système) et les programmes exploitant ces commandes ne seront donc pas compatibles avec les CPC664 et CPC464 + DDI1 sous CP/M 2.2.

TRAITEMENT DE MOTS ET DE LISTES :

(Vous remarquerez les symboles ? et > dans les exemples suivants)

ascii

Donne la valeur ASCII du premier caractère de mot entré.

```
?ascii "G
71
?ascii "g
103
```

bf

(but first = sauf le premier) Donne un objet sans son premier élément.

```
?bf "prendre
rendre
?bf [1 2 3]
[2 3]
```

bl

(but last = sauf le dernier) Donne un objet sans son dernier élément.

```
?bl "prendre
prendr
?bl [1 2 3 4]
[1 2 3]
```

char

Donne le caractère qui correspond à la valeur ASCII entrée.

```
?char 83
S
```

count

Donne le nombre d'éléments de l'objet entré.

```
?count "six
3
?count [0 1 2 3]
4
```

emptyp

Répond **TRUE** (vrai) si l'objet entré et un mot ou une liste vide. Dans le cas contraire répond **FALSE** (faux).

```
?emptyp "
TRUE
?emptyp []
TRUE
?emptyp [x]
FALSE
?make "x []
?emptyp :x
TRUE
```

first

(premier) Donne le premier élément de l'objet entré et supprime les crochets de la liste.

```
?first "zebre
z
?first [1 2 3]
1
```

fput

(first put = met devant) Donne un nouvel objet en ajoutant le premier élément entré juste devant le deuxième.

```
?fput "p "rendre
prendre
?fput 1 [2 3]
[1 2 3]
```

item

(article) Donne l'élément du rang spécifié.

```
?item 4 "manuel
u
```

*
last

Donne le dernier élément de l'objet entré (comparez avec **FIRST**).

```
?last "canibal
l
```

*
lc

Donne le mot entré avec tous les caractères alphabétiques en minuscules (voir également **FIRST**).

```
?lc "GRENIER
grenier
```

list

Donne la liste des éléments entrés en les encadrant de crochets (à comparer avec la primitive **se**).

```
?(list 1 2 3 4)
[1 2 3 4]
?list "peau [rouge]
[peau [rouge]]
?(list)
[]
```

*
listp

Répond **TRUE** (vrai) si l'objet entré est une liste. Dans le cas contraire, répond **FALSE** (faux).

```
?listp "mere
FALSE
?listp [pere frere soeur]
TRUE
```

*
lput

« lastput = mettre en dernier » Donne un nouvel objet en ajoutant le premier élément entré à la suite du deuxième.

```
?lput "s "pluriel
pluriels
?lput "s [pluriel]
[pluriel s]
```

*
memberp

Répond **TRUE** (vrai) si le premier élément entré est un élément du deuxième objet entré.

```
?memberp "y "Elysee
TRUE
?memberp "chocolat [[vanille][chocolat][fraise]]
FALSE
?memberp [chocolat] [[vanille][chocolat][fraise]]
TRUE
```

*
numberp

Répond **TRUE** (vrai) si l'objet entré est un nombre.

```
?numberp 374.926
TRUE
?numberp "six
FALSE
?numberp first [2 4 6 8]
TRUE
```

*
piece

Donne un objet contenant les éléments spécifiés dans l'objet entré.

```
?piece 4 7 "fenetres
etre
?piece 2 4 [francois daniel herve olivier frederic]
[daniel herve olivier]
```

se

(sentence = phrase) Donne la liste des éléments entrés en supprimant les crochets (à comparer avec list).

```
?make "instr_list rl
repeat 4 [fd 50 rt 90]
?run (se "cs :instr_list "ht)
```

Remarque : Le tiret s'obtient en actionnant **[SHIFT]Ø**.

*
shuffle

Donne une liste contenant les éléments de la liste entrée dans un ordre aléatoire.

```
?shuffle [a b c d]
[c b d a]
```

*
uc

Donne le mot entré avec ses caractères alphabétiques en majuscules (comparer avec **lc**).

```
?uc "jones
JONES
```

*
where

Donne un nombre calculé à partir de l'expression memberp vraie la plus récente.

```
?memberp "v "riviere
TRUE
?show where
3
```

word

(mot) Donne un seul mot à partir des mots ou éléments entrés.

```
?word "tele "vision
television
```

wordp

Donne la réponse **TRUE** (vrai) si l'objet entré est soit un mot, soit un nombre.

```
?wordp "bonjour
TRUE
?wordp []
FALSE
```

Opérations arithmétiques :

*
arctan

Donne l'arc-tangente (en degrés) du nombre entré.

```
?arctan 0
0
?arctan 1
45
```

cos

Donne le cosinus du nombre entré en degrés.

```
?cos 60
0.500000000017049
```

int

Donne la partie entière du nombre entré.

```
?int 4/3
1
```

*
quotient

Donne le résultat de la division entière des deux nombres entrés.

```
?quotient 14 4
3
?14/4
3.5
```

random

Donne un nombre entier positif pris au hasard et inférieur au nombre entré.

```
?random 20
```

*
remainder

Donne le résultat entier du quotient du premier nombre entré par le second.

```
?remainder 7 3
1
?remainder 8 4
0
```

*
rerandom

Provoque la répétition d'une expression aléatoire.

```
?repeat 10 [(type random 10 char 9)]
    3   7   5   3   2   0   4   2   6
?repeat 10 [(type random 10 char 9)]
    9   9   1   0   6   1   3   5   1
?rerandom
?repeat 10 [(type random 10 char 9)]
    2   9   0   3   1   6   2   3   7
?rerandom
?repeat 10 [(type random 10 char 9)]
    2   9   0   3   1   6   2   3   7
```

*
round

Donne le nombre entré arrondi à l'entier immédiatement supérieur.

```
?round 3.3333333
3
?round 3.5
4
```

sin

Donne le sinus du nombre entré en degrés.

```
?sin 30
0.500000000017049
```

+

Donne la somme des nombres entrés.

```
?+ 2 2
4
?2+2
4
```

-

Donne la différence des deux nombres entrés.

```
?- 10 5
5
?10-5
5
```

*

Donne le produit des deux nombres entrés.

```
?* 4 6
24
?4*6
24
```

/

Donne le quotient décimal des deux nombres entrés.

```
?/ 25 5
5
?25/5
5
```

Opérations logiques :

and

(et) Répond **TRUE** (vrai) si le résultat de toutes les expressions entrées est vrai.

```
?and (3<4) (7>4)
TRUE
```

not

Répond **TRUE** (vrai) si l'expression entrée est fausse.
Répond **FALSE** (faux) si l'expression entrée est vraie.

```
?not (3=4)
TRUE
?not (3=3)
FALSE
```

or

(ou) Répond **FALSE** (faux) si toutes les expressions entrées sont fausses.

```
?or "TRUE "FALSE
TRUE
?or (3=4) (1=2)
FALSE
```

=

Répond **TRUE** (vrai) si les deux expressions entrées sont équivalentes ; sinon répond **FALSE** (faux).

```
?= "LOGO "LOGO
TRUE
?1=2
FALSE
```

>

Répond **TRUE** (vrai) si le premier élément entré est plus grand que le second ; sinon répond **FALSE** (faux).

```
?> 19 20
FALSE
?20>19
TRUE
```

<

Répond **TRUE** (vrai) si le premier élément entré est plus petit que le second ; sinon répond **FALSE** (faux).

```
?< 27 13
FALSE
?13<27
TRUE
```

Variables :

local

Rend les variables données accessibles uniquement à la procédure en cours.

```
>(local "x "y "z)
```

make

Pour donner une valeur à une variable.

```
?make "cote 50
?:cote
50
```

*

namep

Répond **TRUE** (vrai) si le mot entré identifie une variable définie.

```
?make "gout "chocolat
?:gout
chocolat
?namep "gout
TRUE
?namep "chocolat
FALSE
```

*
thing

Donne la valeur de la variable entrée sous forme de nom

```
?make "ordinateur "amstrad
?thing "ordinateur
amstrad
```

Procédures :

*
define

Construit la procédure définie par la liste et portant le nom spécifié.

```
?define "dire.bonjour [[] [pr "bonjour]]
?po "dire.bonjour
to dire.bonjour
pr "bonjour
.text "dire.bonjour
[[] [pr "bonjour]]
end
```

end

Indique la fin de la procédure. Doit être isolé au début de la dernière ligne de la procédure.

```
?to carre
>repeat 4 [fd 50 rt 90]
>end
carre defined  (carre defini)
?carre
```

po

(print out = affichage) Affiche sur l'écran la procédure désignée ou la valeur des variables spécifiées.

```
?po "carre
to carre
repeat 4 [fd 50 rt 90]
end
?po "x
x is 3   (x est 3)
```

pots

(print out titles = affichage des titres) Affiche les titres de toutes les procédures de l'espace de travail.

```
?pots
```

*
text

Donne la liste de la procédure spécifiée.

```
?to etoile ;cinq branches
>repeat 5 [fd 30 rt 144 fd 30 lt 72]
>end
etoile defined
?text "etoile
[[] [repeat 5 [fd 30 rt 144 fd 30 lt 72]]]
```

to

Indique le début de la construction d'une procédure.

```
?to carre
>repeat 4 [fd 50 rt 90]
>end
carre defined
```

Correction :

ed

Charge sur l'écran la procédure et/ou les variables désignées dans la mémoire d'édition de l'écran.

```
?ed "carre
```

*
edall

Charge sur l'écran toutes les variables et procédures de l'espace de travail désignées dans la mémoire d'édition de l'écran et entre l'éditeur d'écran.

```
?edall
```

*
edf

Charge directement dans la mémoire d'édition de l'écran le fichier spécifié, ou crée un nouveau fichier en entrant l'éditeur d'écran avec une mémoire vide.

```
?edf "etoile
```

Fonctions d'imprimante

*
copyon

Provoque l'impression du texte par écho.

```
?copyon
```

*
copyoff

Arrête l'impression du texte par écho.

```
?copyoff
```

Ecran texte :

ct

(clear text = vidage du texte) Efface tout le texte de la fenêtre où se promène le curseur et ramène le curseur en haut à gauche de celle-ci.

```
?ct
```

*
cursor

Donne les coordonnées du curseur dans la fenêtre texte (numéro de colonne, numéro de ligne).

```
?ct
?cursor
[0 1]
?(type [la  position  actuelle  du  curseur  est\]  show
cursor
la position actuelle du curseur est [32 23]
```

pr

(print = affiche) Affiche les objets entrés au clavier sur l'écran texte, enlève les crochets d'une liste et va à la ligne (à comparer avec **show** et **type**).

```
?pr [a b c]
a b c
```

*

setcursor

Place le curseur aux coordonnées spécifiées.

```
?ct
?to dessin
>make "x random 20
>make "y random 12
>setcursor list :x :y pr "*
>end
?dessin
```

setsplit

Délimite le nombre de lignes de l'écran texte.

```
?setsplit 10
```

show

Affiche les objets entrés au clavier sur l'écran texte, garde les crochets extérieurs d'une liste et va à la ligne (à comparer avec **pr** et **type**).

```
?show [a b c]
[a b c]
```

ts

(text screen = écran texte) Réserve tout l'écran pour le texte.

```
?ts
```

type

Affiche les objets entrés au clavier sur l'écran texte, enlève les crochets d'une liste et ne va pas à la ligne (à comparer avec **pr** et **show**).

```
?type [a b c]
a b c
```

L'écran graphique :

Notez que l'écran est en mode 1, à 4 couleurs, et que le système de coordination est le même que celui du BASIC AMSTRAD. En d'autres termes, chaque position sur l'écran sera arrondie au point écran ayant le numéro pair le plus proche. Les quantités de rouge, de vert et de bleu pourront varier entre 0, 1 ou 2.

clean

(to clean = nettoyer) Efface tout l'écran graphique sans toucher à la tortue.

```
?fd 50
?clean
```

cs

(clear screen = vide l'écran) Efface l'écran graphique et ramène la tortue en position [0,0] tournée vers 0 (le nord) et avec le stylo baissé.

```
?rt 90 fd 50
?cs
```

dot

Place un point sur l'écran à la position spécifiée par les coordonnées indiquées et selon la couleur de stylo en cours.

```
?dot [50 10]
```

*
dotc

Donne le numéro de la couleur du point spécifié par les coordonnées, ou -1 si le point n'est pas à l'écran.

```
?cs
?setpc 1
?dot -50 50
?setpc 2
?dot 50 50
?setpc 3
?dot 50 -50
?dotc 50 50
2
?dotc -50 -50
0
?dotc 1000 3000
-1
```

fence

(barrière) Pose une barrière qui empêche la tortue de sortir des limites de l'écran. **window** enlève cette barrière.

```
?fence
?fd 300
Turtle out of bounds (la tortue est hors des limites)
```

*
fill

Peint une zone dans la couleur du stylo en cours en changeant le point situé sous la tortue et tous les points contigus horizontaux et verticaux en utilisant l'état du stylo en cours.

```
?make "x 5
?cs
?st
?pd
?repeat 30 [fd :x rt 90 make "x :x + 5]
?fd 20 rt 90
?fd 10
?pu
?home
?bk 2
?pd
?setpc 2
?fill
```

fs

(full screen = plein écran) Réserve la totalité de l'écran au graphique.

```
?fs
```

pal

(palette) Donne les trois nombres représentant les quantités de rouge, vert, bleu du stylo.

```
?pal 2
[0 2 2]
```

*
setbg

Change la couleur de fond de l'écran graphique en fonction du numéro de couleur spécifié.

```
?sf
[0 SS 5 FENCE 1]
```

(cette instruction définit le fond à la couleur 0)

```
?pal 0
[0 0 1]
?setbg 2
?sf
[2 SS 5 FENCE 1]
```

setpal

(set palette = fabrication des couleurs) Les trois chiffres assignent les quantités de rouge, de vert, de bleu du stylo.

```
?setpal 3 [1 1 2]
?pal 3
[1 1 2]
```

*
setscrunch

Donne au rapport de l'écran graphique la valeur du nombre spécifié.

```
?sf
[0 SS 5 FENCE 1]
?to cercle
>repeat 360 [fd 1 rt 1]
>end
cercle defined
?setscrunch 2
?sf
[0 SS 5 FENCE 2]
?cercle
?setscrunch 2.5
?cercle
```

sf

(screen facts = les données de l'écran) Affiche des informations sur l'état de l'écran graphique. La structure de ces informations est :[<couleur de fond> <état de l'écran> <longueur de l'écran texte> <fenêtre> <scrunch>] où <couleur de fond> est la couleur du stylo de fond (toujours 0 sous CP/M 2.2). <état de l'écran> indique **SS** (split screen = écran partagé), **FS** (full screen = plein écran) ou **TS** (text screen = écran totalement réservé au texte). <longueur de l'écran texte> est le nombre de lignes réservé au texte, et <fenêtre> indique **WINDOW** (pas de barrière, la tortue peut sortir), **WRAP** (réapparition de la tortue de l'autre côté de l'écran) ou **FENCE** (barrière - la tortue ne peut pas sortir des limites de l'écran). <scrunch>, le rapport d'écran (non disponible sous CP/M 2.2), est toujours égal à 1 par défaut et peut être modifié par **SETSCRUNCH**.

```
?sf
[0 SS 5 FENCE 1]
```

ss

(split screen = écran partagé) Réserve au texte une fenêtre sur l'écran graphique.

```
?ss
```

window

Permet à la tortue de sortir des limites de l'écran après un wrap ou un fence.

```
?fence fd 300
Turtle out of bounds
?window
?fd 300
```

wrap

Fait apparaître la tortue du côté opposé à celui où elle sort.

```
?cs wrap
?rt 5 fd 1000
? cs window
?rt 5 fd 1000
```

Les graphismes de la tortue :

bk

(back = arrière) Déplace la tortue du nombre de pas indiqué dans la direction opposée au sens de la flèche.

```
?cs fd 150
?bk 50
```

fd

(forward = avant) Avance la tortue du nombre de pas indiqué dans le sens de la flèche.

```
?fd 80
```

*
Home

Replace la tortue au centre de l'écran graphique, position [O O], tournée vers 0 (le nord).

```
?fd 100
?rt 45
?fd 100
?home
```

ht

(hide turtle = cache tortue) Rend la tortue invisible. Accélère et clarifie le dessin.

```
?ht
?cs fd 50
?st
```

lt

(left = gauche) La tortue tourne du nombre de degrés indiqué vers la gauche.

```
?lt 90
```

pd

(pen down = stylo baissé) Le stylo est posé sur le papier et la tortue peut recommencer à tracer après interruption par la primitive pu.

```
?fd 20 pu fd 20
?pd
?fd 20
```

pe

(pen erase = stylo qui efface) Cette instruction transforme la couleur du stylo en 0, c'est-à-dire la couleur du fond, et permet ainsi d'effacer les traits sur l'écran.

```
?fd 50
?pe
?bk 25
?fd 50
?pd fd 25
```

pu

(pen up = stylo levé) Le stylo est levé, ne laissant plus de trace sur l'écran.

```
?fd 30
?pu
?fd 30
?pd fd 30
```

px

(pen exchange = stylo inversé) Change la couleur de tout ce qui a été tracé auparavant en sa couleur opposée ou logiquement complémentaire.

```
?fd 20 pu fd 20
?pd setpc 3 fd 20
?px
?bk 80
?fd 80
?pd bk 100
```

rt

(right = droite) La tortue tourne du nombre de degrés indiqué vers la droite.

```
? rt 90
```

seth

(set heading = changement de direction) Tourne la tortue dans la direction spécifiée en degrés. Les nombres positifs font tourner la tortue dans le sens des aiguilles d'une montre, les nombres négatifs en sens inverse.

```
?seth 90
```

setpc

(set pen coulour = changement de la couleur du stylo) Le stylo prend la couleur indiquée par le nombre donné (0 est la couleur du fond).

```
?setpc 1
```

setpos

(set position = changement de position) Envoie la tortue à la position indiquée par les coordonnées spécifiées.

```
?setpos [30 20]
```

*
setx

Change la position horizontale de la tortue en fonction de la coordonnée x spécifiée (voir également **sety**).

```
?setx 80
?fd 100
?setx -50
?fd 50
```

*
sety

Change la postion verticale de la tortue en fonction de la coordonnée y spécifiée.

```
?sety 90
?fd 20
?sety -50
?fd 50
```

st

(show turtle = montre tortue) Pour rendre visible à nouveau une tortue invisible.

```
?ht
?fd 50
?st
```

tf

(turtle facts = données de la tortue) Affiche une liste d'informations concernant la tortue. Le format est : [<coordx> <coordy> <direction> <état du stylo> <couleur du stylo n> <visibilité>] où <coordx> est la coordonnée x de la tortue ; <coordy> est la coordonnée y de la tortue ; <direction> indique la direction vers laquelle se tourne la tortue (en degrés) ; <visibilité> est **TRUE** (vrai) lorsque la tortue est visible ; <état du stylo> indique **PD** (pen down = stylo baissé), **PE** (pen erase = stylo qui efface), **PX** (pen exchange = stylo inversé), **PU** (pen up = stylo levé) ; <couleur du stylo n> identifie le numéro de la couleur du stylo n.

```
?setpos [15 30]
?rt 60
? setpc 3
?pe
?ht
?tf
[15 30 60 PE 3 FALSE]
```

*
towards

Donne une direction qui pointe la tortue vers les coordonnées spécifiées.

```
?seth towards list :x :y
```

Gestion de l'espace de travail :

er

(erase = efface) Efface la procédure spécifiée de l'espace de travail.

```
?er "carre
```

*
erall

Efface toutes les procédures et les variables de l'espace de travail.

```
?erall
```

ern

(erase name = efface le nom) Efface la ou les variables spécifiées de l'espace de travail.

```
?make "cote [100]
?make "angle [45]
?:cote :angle
[100]
[45]
?ern [cote angle]
?:cote
cote has no value  (cote ne vaut rien)
```

nodes

Affiche le nombre de nœuds disponible dans l'espace de travail.

```
?nodes
```

*
noformat

Supprime de l'espace de travail le formatage de la procédure et les commentaires afin de libérer les nœuds.

```
?noformat
```

*
poall

Affiche les définitions de toutes les procédures et variables de l'espace de travail.

```
?poall
```

*
pons

Affiche les noms et les valeurs de toutes les variables globales de l'espace de travail.

```
?pons
medium is 40
small is 20
large is 80
```

*
pops

Affiche les noms et les définitions de toutes les procédures de l'espace de travail.

```
?pops
```

recycle

Libère le plus de nœuds possible et réorganise l'espace de travail.

```
?recycle
?nodes
```

Listes de propriété :

glist

(get list = donne la liste) Permet d'obtenir une liste de tous les objets de l'espace de travail auxquels on a attaché une ou des propriétés.

```
?glist ".DEF
```

gprop

(get property = donne la propriété) Permet d'obtenir la valeur de la propriété du nom désigné.

```
?make "hauteur "72
?gprop "hauteur ".APV
72
```

plist

(property list = liste des propriétés) Permet d'obtenir la liste des propriétés attachées à un mot.

```
?plist "hauteur
[.APV 72]
```

pprop

(put property = donne une propriété) Permet de créer un lien entre un mot et une propriété.

```
?pprop "maitre ".APV "Corbeau
?:maitre
Corbeau
```

*

pps

Affiche les paires de propriétés non standard de tous les objets de l'espace de travail.

```
?pprop "Helene "telephone 2132256
?pps
Helene's telephone is 213
?plist "Helene
[telephone 2132256]
```

remprop

(remove property = élimine la propriété) Elimine la propriété spécifiée de la liste de propriété du mot.

```
?remprop "maitre ".APV
```

Fichiers sur disquette :

*
changef

Change le nom d'un fichier du catalogue de la disquette.

```
?dir
[CERCLE CARRE ETOILE]
?changef "boite "carre
?dir
[CERCLE BOITE ETOILE]
```

*
defaultd

Donne le nom de l'unité par défaut en cours.

```
?defaultd
A:
```

dir

(directory = catalogue) Permet d'obtenir une liste de noms de fichiers en Dr. LOGO présents sur une disquette, soit spécifiée, soit prise par défaut ; accepte également les jockers.

```
?dir "a:????????
```

(Etudiez la partie 1 du chapitre intitulé « AMSDOS et CP/M » pour l'utilisation des jockers???????? Dr. LOGO ne tient toutefois pas compte du jocker *.)

*
dirpic

Donne une liste des noms de fichiers graphiques de l'unité par défaut ou spécifiée. Accepte les noms de fichiers ambigus.

```
?dirpic "b:
[MON_DESS CARRES ETOILES CERCLES]
```

load

Charge le fichier spécifié, de la disquette sur l'espace travail de l'ordinateur.

```
?load "monfichier
?load "b:dessins
```

*
loadpic

Recrée sur l'écran graphique le dessin sauvegardé dans le fichier graphique spécifié.

```
?loadpic "mon_dess
?loadpic "b:mon_dess
```

save

Ecrit le contenu de l'espace travail dans un fichier sur la disquette spécifiée.

```
?save "dessins
```

Remarque

Avant de sauvegarder, insérez une disquette formatée disposant d'une capacité suffisante. Ne sauvegardez pas sur une disquette système. Il est préférable de fermer le trou de protection en écriture pour éviter tout risque d'écriture accidentelle.

Si vous utilisez la version CP/M 2.2 du Dr. LOGO (face 4 d'une des disquettes système), veillez à ce que votre disquette de travail dispose de l'espace nécessaire au stockage des programmes, car vous ne pouvez pas changer de disquette en cours de procédure.

*
savepic

Ecrit le contenu de l'écran graphique dans le fichier graphique spécifié.

```
?savepic "mon_dess
?savepic "b:mon_dess
```

Remarque

Avant de sauvegarder, insérez une disquette formatée disposant d'une capacité suffisante. Ne sauvegardez pas sur une disquette système. Il est préférable de fermer le trou de protection en écriture pour éviter tout risque d'écriture accidentelle.

Si vous utilisez la version CP/M 2.2 du Dr. LOGO (face 4 d'une des disquettes système), veillez à ce que votre disquette de travail dispose de l'espace nécessaire au stockage des programmes, car vous ne pouvez pas changer de disquette en cours de procédure.

*
setd

(setdrive) Définit l'unité spécifiée comme unité par défaut.

```
?defaultd
A:
?dir
[BOITE CERCLE TRIANGLE]
?setd b:
?defaultd
B:
?dir
[ETOILE MAISON]
```

Clavier et manette de jeu :

buttonp

(button pressed = bouton appuyé) Répond **TRUE** (vrai) si le bouton de la manette de jeu est actionné. Les numéros 0 et 1 servent à identifier les deux manettes de jeu.

```
?to tirer
>label "boucle
>if (buttonp 0) [pr [tirer 0!]]
>if (buttonp 1) [pr [tirer 1!]]
>go "boucle
>end
```

La position de la manette de jeu est testée par **paddle**.

keyp

Répond **TRUE** (vrai) si un caractère vient d'être tapé au clavier et attend d'être enregistré.

```
?to inkey
>if keyp [op rc] [op "]
>end
```

paddle

Permet de connaître les états de la manette 0 ou 1. Les différentes positions sont codées comme suit :

Valeur retournée	Signification
255	Rien
0	En haut
1	En haut et à droite
2	A droite
3	En bas et à droite
4	En bas
5	En bas et à gauche
6	A gauche
7	En haut et à gauche

```
?paddle 0
255
```

Les boutons de tir sont testés par la commande buttonp.

rc

(read character = sortie d'un caractère) Sort le premier caractère tapé au clavier.

```
?make "touche rc
```

...et appuyez sur la touche X :

```
?:touche
X
```

rl

(read list = sortie d'une liste) Donne une liste contenant une entrée au clavier qui doit être suivie d'un retour chariot.

```
?make "instr_list rl
repeat 4 [fd 50 rt 90]
?:instr_list
[repeat 4 [fd 50 rt 90]]
```

rq

(read quote = extraction d'un mot) Donne un mot contenant une ligne entrée au clavier, qui doit être suivi d'un retour chariot.

```
?make "commande rq
repeat 3 [fd 60 rt 120]
?:commande
repeat 3 [fd 60 rt 120]
```

LE SON :

Seule la version Schneider de Dr. LOGO possède des commandes pour le son. Elles sont d'ailleurs identiques à leurs équivalents BASIC. Reportez-vous à la partie 9 du Cours Elémentaire pour plus de détails.

sound

Met un son dans une suite de sons (queue). Le format de cette instruction est : [<état de canal> <période sonore> <durée> <volume> <enveloppe de volume> <enveloppe de tonalité> <bruit>]. Tous les paramètres après durée sont facultatifs.

```
?sound [1 20 50]
```

env

Construit l'enveloppe de volume (la forme de la note). Son format est :
[<numéro de l'enveloppe> <section(s) de l'enveloppe>]

```
?env [1 100 2 20]
?sound [1 200 300 5 1]
```

ent

Construit l'enveloppe de tonalité. Son format est : [<numéro de l'enveloppe> <section(s) de l'enveloppe>]

```
?ent [1 100 2 20]
?sound [1 200 300 5 1 1]
```

release

Libère les canaux dont l'exécution a été suspendue par la commande sound. Les canaux à libérer sont indiqués par le code suivant :

Valeur entrée	Canaux libérés
Ø	Aucun
1	A
2	B
3	A et B
4	C
5	A et C
6	B et C
7	A et B et C

```
?release 1
```

Les commandes de déroulement de programme :

bye

Lorsque l'on en a terminé avec Dr.LOGO.

```
?bye
```

co

Pour continuer après une pause provoquée par **[CTRL]Z**, **pause** ou **ERRACT**.

```
?co
```

go

Exécute la ligne de commande repérée par une étiquette à l'intérieur d'une procédure.

```
>go "boucle
```

if

Exécute une ou deux listes d'instructions suivant la valeur de l'expression entrée ; ces listes en toutes lettres doivent être entourées de crochets.

```
>if (:a>:b) [pr [a est plus grand]]
>if (:a<:b) [pr [b est plus grand]]
```

label

Pose une étiquette sur une ligne d'instruction qui pourra ainsi être appelée à l'aide de la commande go.

```
>label "boucle
```

op

(output = retourne) Affiche l'objet entré et interrompt la procédure.

```
?op [resultat]
```

repeat

Répète la liste d'instructions autant de fois que le numéro le spécifie.

```
?repeat 4 [fd 50 rt 90]
```

run

Exécute la ligne d'instructions.

```
?make "instr_list [fd 40 rt 90]
?run :instr_list
```

stop

Interrompt l'exécution de la procédure en cours et revient au niveau supérieur **TOPLEVEL** (le symbole ?) ou à la procédure d'appel.

```
?stop
```

wait

Arrête l'exécution de la procédure pendant le temps spécifié par le nombre entré.

La longueur du temps de pause est : nombre entré * 1/60 seconde.

```
?wait 20
```

Traitements des cas particuliers :

catch

(attrape) Pour traiter une erreur ou un cas particulier qui survient pendant l'exécution d'une série d'instructions.

```
>catch "error [+ [] []]
>pr [je suis ici]
je suis ici
```

error

Donne la liste d'instructions qui a causé la dernière erreur.

```
>catch "error [execute jusqu'a l'erreur]
>show error
```

*
notrace

Désactive **trace** (voir cette commande).

```
?notrace
```

*
nowatch

Désactive **watch** (voir cette commande).

```
?nowatch
```

pause

Suspend l'exécution de la procédure en cours, permettant ainsi le dialogue entre Dr. LOGO et l'utilisateur.

```
>if :cote>5 [pause]
```

throw

Exécute la ligne d'instruction identifiée par le nom entré sous **catch**.

```
?throw "TOPLEVEL
```

*
trace

Affiche le nom de chaque procédure en cours d'exécution.

```
?trace
```

*
watch

Affiche le nom de chaque expression en cours d'exécution.

```
?watch
```

Les primitives système :

.contents

Affiche le contenu de l'emplacement des symboles de Dr. LOGO.

.deposit

Place le deuxième nombre entré à l'emplacement mémoire donné par le premier nombre.

.examine

Affiche le contenu de l'emplacement mémoire spécifié.

*
.in

Recherche la valeur en cours de l'entrée spécifiée.

*
out

Envoie la valeur introduite vers l'entrée spécifiée.

Les variables système :

ERRACT

Renvoie au niveau supérieur **TOPLEVEL**, lorsque **TRUE** (vrai) provoque une pause au moment d'une erreur.

FALSE

Valeur système.

REDEFP

Quand **TRUE** (vrai) permet la redéfinition des primitives.

TOPLEVEL

throw"TOPLEVEL clos toutes les procédures en attente.

TRUE

Valeur système.

Les propriétés du système :

.APV

Valeur de la variable associée à la propriété ; à savoir la valeur de la variable globale.

.DEF

Définition d'une procédure

*
.ENL

Fin d'une ligne de procédure interrompue par un retour chariot ou des espaces.

*
.EMT

Début d'une ligne de procédure interrompue par un retour chariot ou des espaces.

.PRM

Identifie une primitive.

*
.REM (ou;)

Remarques ou commentaires.

Chapitre 7
Pour information...

Ce chapitre fournit les informations de référence susceptibles de vous aider à utiliser votre ordinateur.

Sujets abordés :

* Déplacements du curseur et fonctions des codes de contrôle
* Interruptions
* Caractères graphiques et ASCII
* Références clés
* Sons
* Messages d'erreur
* Mots clés du BASIC
* Grilles
* Connexions
* Imprimantes
* Manettes de jeu
* Organisation des disques
* Mémoire
* Organisation des disques
* Extensions résidentes du système (RSX)
* Mémoire
* Emulateur de terminaux sous CP/M Plus
* Jeu de caractères sous CP/M Plus

Pour des informations complètes sur le BASIC et les microprogrammes du 6128, consultez respectivement les manuels SOFT 967 et SOFT 968 d'AMSOFT.

Partie 1 : Déplacements du curseur et fonctions des codes de contrôle

Dans un certain nombre de progiciels (ou programmes d'application), le curseur de texte peut se trouver à l'extérieur de la fenêtre utilisée. Certaines opérations commencent par ramener le curseur à sa position légale. Ce sont :

1. L'écriture d'un caractère

2. Le dessin du bloc du curseur

3. L'envoi d'un code de contrôle marqué d'un astérisque dans les pages suivantes.

La procédure qui ramène le curseur à une position légale est la suivante :

1. Si le curseur est complètement à droite, il est ramené à l'extrême gauche de la ligne suivante.

2. S'il se trouve à gauche du bord gauche, il est ramené à droite sur la ligne précédente.

3. Si le curseur est au-dessus du bord supérieur, l'ensemble de la fenêtre descend alors d'une ligne et le curseur se place sur la première ligne.

4. Si le curseur est en dessous du bord inférieur, l'ensemble de la fenêtre monte alors d'une ligne et le curseur se place sur la dernière ligne.

Les tests et opérations se font dans l'ordre indiqué. Les positions de curseur illégales peuvent s'exprimer par un nombre nul ou négatif, correspondant à la position à l'extérieur du bord gauche ou au-dessus d'une fenêtre. Les valeurs de caractère de 0 à 31 ne produisent rien sur l'écran et ne devraient pas être utilisées inconsidérément. Certains codes changent la signification du ou des caractères qui les suivent, les considérant comme paramètres d'une fonction particulière.

S'il est généré à partir du clavier, un code de contrôle envoyé à l'écran graphique provoque simplement l'affichage du symbole conventionnel correspondant à sa fonction (exemple : &07'BEL' - **[CTRL]G**). Il ne remplit sa fonction de contrôle qu'envoyé par l'une des commandes suivantes :

PRINT CHR$ (&Ø7), ou **PRINT** "␇ " (␇s'obtient en appuyant sur **[CTRL]G** à l'intérieur de l'instruction **PRINT**).

Les codes marqués d'un astérisque * commencent par forcer le curseur à se rendre sur une position légale - mais peuvent laisser le curseur dans une position non légale. Les codes sont indiqués d'abord avec leur valeur hexadécimale (&XX) puis l'équivalent décimal.

Codes de contrôle

Valeur	Nom	Paramètre	Fonction
&00 0	NUL		Pas d'effet. Ignoré.
&01 1	SOH	0 à 255	Affiche le symbole correspondant à la valeur du paramètre. Permet ainsi l'affichage des symboles de 0 à 31.
&02 2	STX		Supprime le curseur de texte. Equivaut à la commande **CURSOR** accompagnée d'un paramètre utilisateur égal à 0.
&03 3	ETX		Active le curseur de texte. Equivaut à la commande **CURSOR** accompagnée d'un paramètre utilisateur égal à 1. Pour afficher un curseur à l'intérieur d'un programme en BASIC (autre que le curseur automatique généré par le BASIC en attente d'une entrée au clavier), il faut faire appel à une commande **CURSOR** avec un paramètre système égal à 1.
&04 4	EOT	0 à 2	Active le Mode écran. Valeur du paramètre modulo 4. Equivaut à une commande **MODE**.
&05 5	ENQ	0 à 255	Envoie le caractère codé par le paramètre à la position du curseur graphique.
&06 6	ACK		Active l'écran de texte (voir &15 NAK).
&07 7	BEL		Fait sonner la cloche. (Vide les files d'attente des canaux sonores).
&08 8 *	BS		Déplace le curseur d'une colonne à gauche.

Valeur	Nom	Paramètre	Fonction
&09 9 *	TAB		Déplace le curseur d'une colonne à droite.
&0A 10 *	LF		Déplace le curseur d'une ligne vers le bas.
&0B 11 *	VT		Déplace le curseur d'une ligne vers le haut.
&0C 12	FF		Efface l'écran de texte et place le curseur en haut à gauche ; équivaut à une commande **CLS**.
&0D 13 *	CR		Amène le curseur jusqu'au bord gauche de la fenêtre sur la ligne courante.
&0E 14	SO	0 à 15	Définit l'Encre du Papier = Commande **PAPER**. Valeur du paramètre modulo 16.
&0F 15	SI	0 à 15	Définit l'Encre du stylo = Commande **PEN**. Valeur du paramètre modulo 16.
&10 16 *	DLE		Efface le caractère. Remplit la matrice du caractère avec la couleur du Papier.
&11 17 *	DC1		Efface depuis le bord gauche de la fenêtre jusqu'au caractère courant, en remplissant les matrices de caractère concernées avec l'encre du Papier.
&12 18 *	DC2		Efface à partir du caractère courant (inclus) jusqu'au bord droit de la fenêtre. Remplit les matrices de caractères concernées avec l'encre du papier.
&13 19 *	DC3		Efface depuis le début de la fenêtre jusqu'au caractère courant. Remplit avec l'encre du Papier.

Valeur	Nom	Paramètre	Fonction
&14 20 *	DC4		Efface à partir du caractère jusqu'à la fin de la fenêtre. Remplissage avec l'encre du Papier.
&15 21	NAK		Désactive l'écran de texte. Rien ne s'affichera jusqu'à ce qu'un caractère **ACK** (&066) ait été envoyé.
&16 22	SYN	0 à 1	Paramètre modulo 2. 0 supprime le mode transparent, 1 établit le mode transparent.
&17 23	ETB	0 à 3	Paramètre modulo 4. 0 valide le mode d'Encre Graphique NORMAL 1 valide le mode d'Encre Graphique XOR (ou exclusif) 2 valide le mode d'Encre Graphique AND (et) 3 valide le mode d'Encre Graphique OR (ou inclusif)
&18 24	CAN		Echange des encres papier et stylo.
&19 25	EM	0 à 255 0 à 255 0 à 255 0 à 255 0 à 255 0 à 255 0 à 255 0 à 255 0 à 255	Equivaut à la commande **SYMBOL**. Admet neuf paramètres dont le premier concerne le code du caractère ; les suivants correspondent chacun à une ligne de la matrice des caractères. Le bit de poids le plus fort du 1er octet correspond au point supérieur gauche du caractère, tandis que le plus faible correspond à son point inférieur droit.
&1A 26	SUB	1 à 80 1 à 80 1 à 25 1 à 25	Etablit la fenêtre = commande **WINDOW**. Les deux premiers paramètres définissent les bords gauche et droit, la plus petite valeur correspondant au bord gauche et la plus grande au bord droit. Les deux derniers paramètres définissent les bords supérieur et inférieur. La plus petite valeur correspond au bord supérieur et la plus grande au bord inférieur.

Valeur	Nom	Paramètre	Fonction
&1B 27	ESC		Pas d'effet. Ignoré.
&1C 28	FS	0 à 15 0 à 31 0 à 31	Attribue deux couleurs à une encre. Equivaut à une commande **INK**. Le premier paramètre définit le numéro de l'encre (modulo 16), les suivants définissent les couleurs (modulo 32). Les valeurs des paramètres 27 à 31 correspondent à des couleurs non définies.
&1D 29	GS	0 à 31 0 à 31	Attribue deux couleurs au cadre (**border**). Equivaut à la commande **BORDER**. Les deux paramètres (modulo 32) spécifient les deux couleurs. Les valeurs 27 et 31 sont indéfinies.
&1E 30	RS		Amène le curseur vers le coin supérieur gauche de la fenêtre.
&1F 31	US	1 à 80 1 à 25	Amène le curseur à une position donnée dans la fenêtre courante. Equivaut à une commande **LOCATE**. Le premier paramètre spécifie la colonne, le deuxième spécifie la ligne.

L'organisation interne du 6128 est assurée par un système d'exploitation perfectionné travaillant en temps réel. Le système d'exploitation gère le « trafic des informations » dans l'ordinateur depuis les entrées jusqu'aux sorties.

Il sert principalement de lien entre la machine et l'interpréteur BASIC - pour les couleurs clignotantes, par exemple, le BASIC passe les paramètres au système d'exploitation qui organise le travail. L'un détermine ce qu'il faut faire et l'autre quand il faut le faire.

Le système d'exploitation fait partie du logiciel intégré à la machine (que les anglais appellent firmware, c'est-à-dire mélange de hardware et de software). Il contient les routines en code machine appelées par le BASIC.

Si vous désirez expérimenter les commandes **POKE** et **CALL**, commencez par sauvegarder et lister votre programme, car vous risqueriez de regretter vos manœuvres. Dépassant l'objectif de cet ouvrage, le système d'exploitation du logiciel est décrit en détail dans le manuel SOFT 946.

Pour programmer en code machine, vous devez faire appel à un assembleur. L'assembleur DEVPAC d'AMSOFT comprend un assembleur Z80 translatable, un éditeur, un désassembleur et un moniteur.

Partie 2 : Interruptions de programme

Pour implémenter plusieurs informations multi-tâches, le 6128 fait appel aux interruptions du Z80, dont **AFTER** et **EVERY** sont des exemples. Voici l'ordre de priorité des compteurs d'événement :

Rupture **([ESC] [ESC])**
Compteur 3
Compteur 2 (et les 3 files d'attente des canaux sonores)
Compteur 1
Compteur 0

Les interruptions doivent tenir compte des états intermédiaires d'une variable. Le sous-programme d'interruption doit éviter les interactions indésirables avec l'état des variables du programme principal.

Les files d'attente des canaux sonores ont des interruptions de même priorité. Ainsi, lorsque le sous-programme d'interruption d'une file d'attente d'un canal sonore s'exécute, il ne peut être interrompu par aucune autre file d'attente. Ceci permet aux sous-programmes d'interruption de partager des variables en évitant les incidences fâcheuses que nous venons d'évoquer.

Quand une interruption propre à la file d'attente d'un canal sonore a été autorisée (à l'aide de **ON SQ GOSUB**), elle est immédiatement prise en compte si la file concernant le canal n'est pas pleine. Dans le cas contraire, elle se produit à la fin de la génération sonore courante (s'il reste de la place dans la file). Le fait d'être interrompu, dévalide l'événement de sorte que le sous-programme d'interruption doit procéder à la réautorisation pour que d'autres interruptions soient susceptibles d'intervenir.

La tentative de générer un son et de tester l'état de la file a également pour conséquence de dévalider une interruption.

Partie 3 : Caractères ASCII et graphiques

ASCII

A titre de référence, nous reproduisons ici le jeu de caractères standard ASCII avec la notation décimale, octale et hexadécimale. Les pages suivantes donnent le détail de chaque matrice de caractères du 6128.

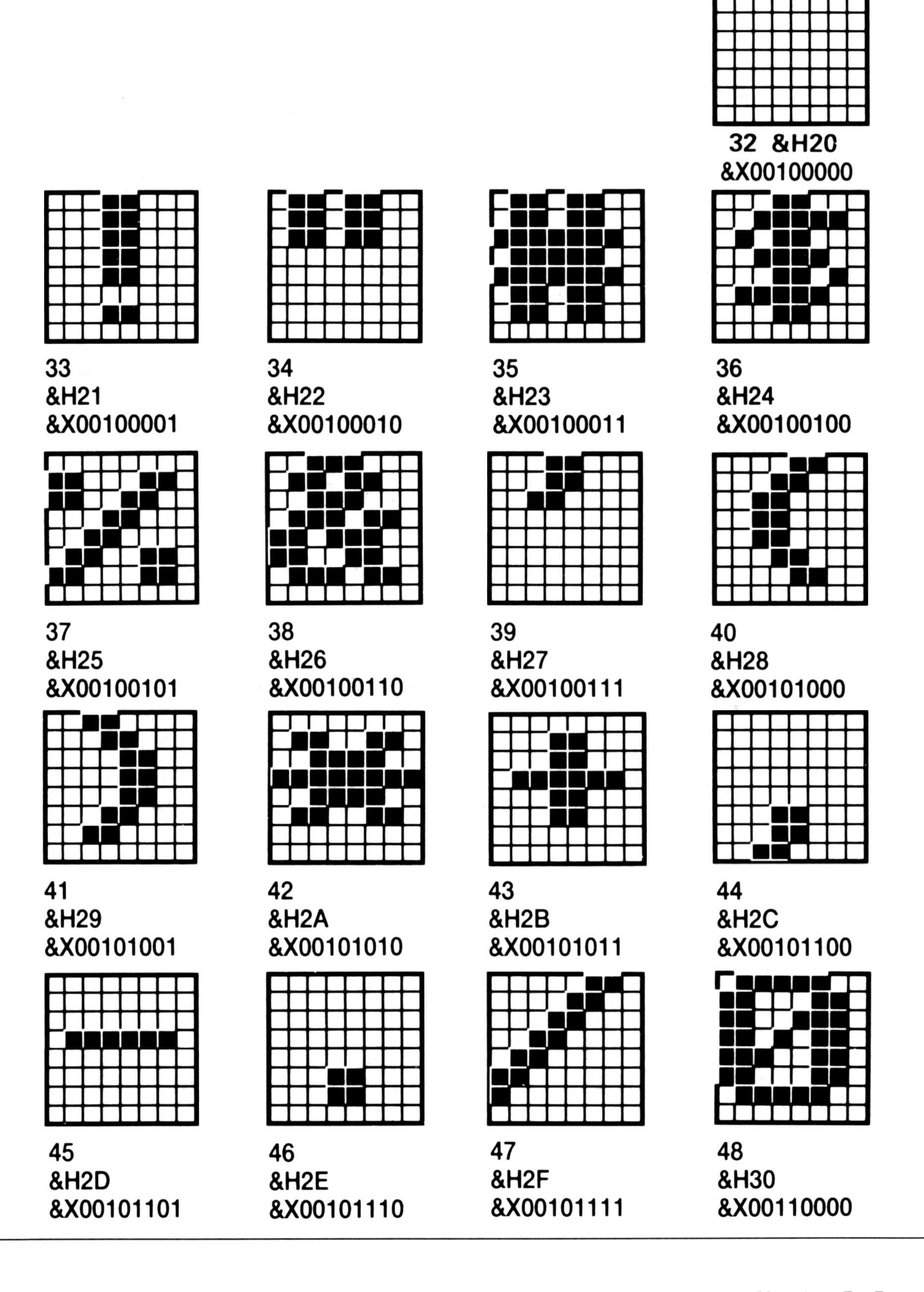
DÉCI.	OCTAL	HEXA.	Caractères ASCII
0	000	00	NUL ([CTRL]@)
1	001	01	SOH ([CTRL]A)
2	002	02	STX ([CTRL]B)
3	003	03	ETX ([CTRL]C)
4	004	04	EOT ([CTRL]D)
5	005	05	ENQ ([CTRL]E)
6	006	06	ACK ([CTRL]F)
7	007	07	BEL ([CTRL]G)
8	010	08	BS ([CTRL]H)
9	011	09	HT ([CTRL]I)
10	012	OA	LF ([CTRL]J)
11	013	OB	VT ([CTRL]K)
12	014	OC	FF ([CTRL]L)
13	015	OD	CR ([CTRL]M)
14	016	OE	SO ([CTRL]N)
15	017	OF	SI ([CTRL]O)
16	020	10	DLE ([CTRL]P)
17	021	11	DC1 ([CTRL]Q)
18	022	12	DC2 ([CTRL]R)
19	023	13	DC3 ([CTRL]S)
20	024	14	DC4 ([CTRL]T)
21	025	15	NAK ([CTRL]U)
22	026	16	SYN ([CTRL]V)
23	027	17	ETB ([CTRL]W)
24	030	18	CAN ([CTRL]X)
25	031	19	EM ([CTRL]Y)
26	032	1A	SUB ([CTRL]Z)
27	033	1B	ESC
28	034	1C	FS
29	035	1D	GS
30	036	1E	RS
31	037	1F	US
32	040	20	SP
33	041	21	!
34	042	22	"
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	'
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	,
45	055	2D	-
46	056	2E	.
47	057	2F	/
48	060	30	Ø
49	061	31	1

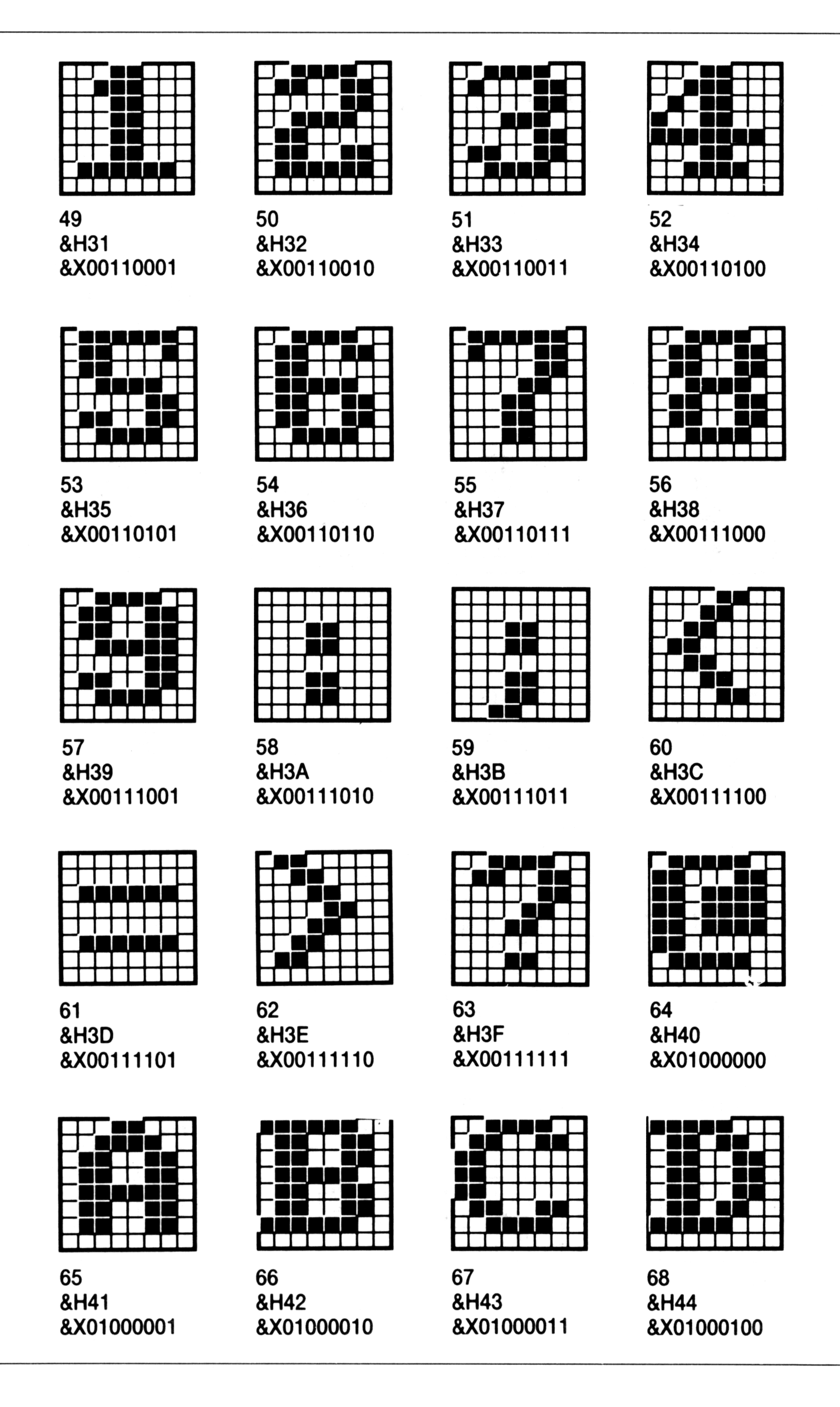
DÉCI.	OCTAL	HEXA.	ASCII
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=
62	076	3E	>
63	077	3F	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	↑
95	137	5F	–
96	140	60	`
97	141	61	a
98	142	62	b
99	143	63	c

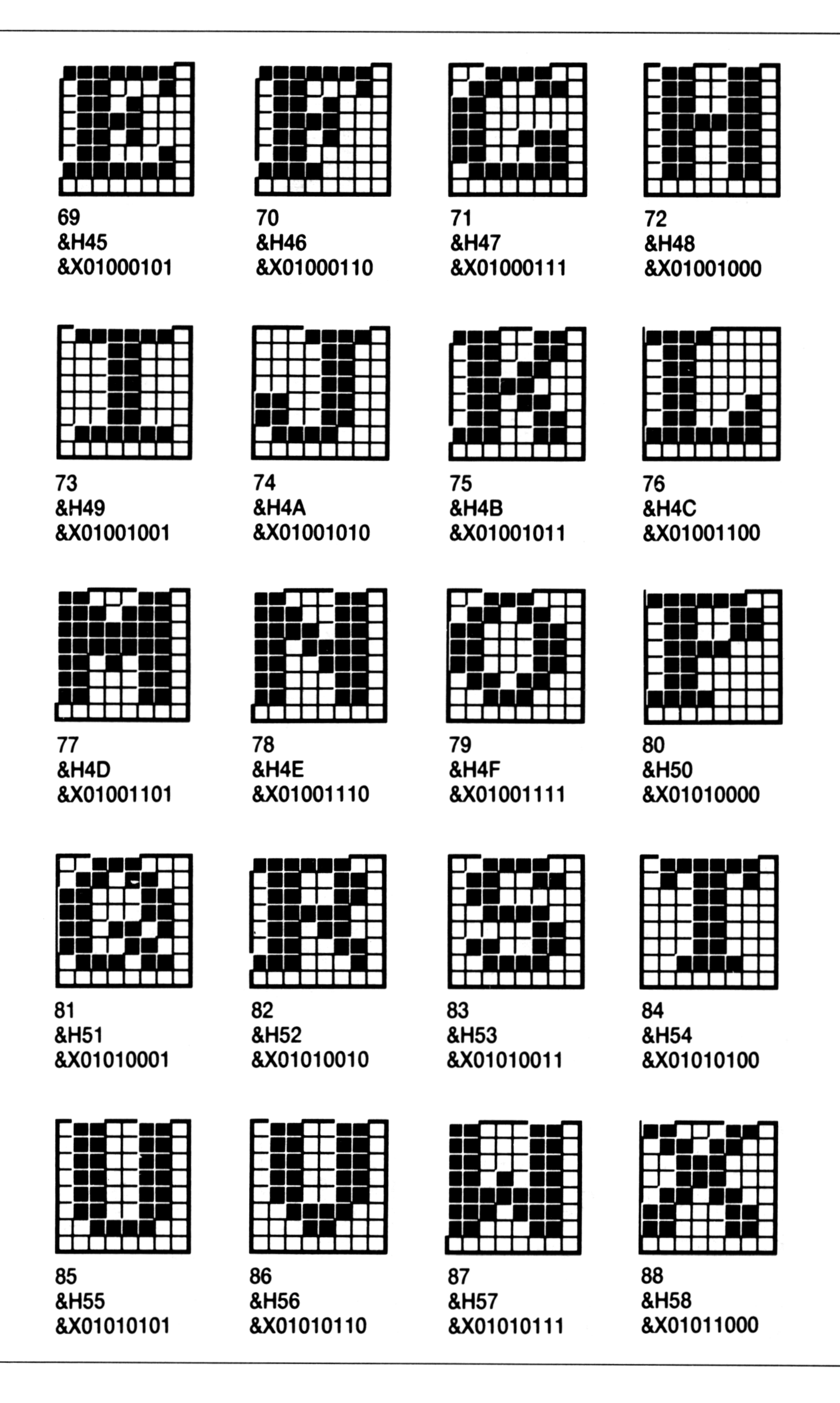
DÉCI.	OCTAL	HEXA.	ASCII
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~

Le jeu de caractères particulier du 6128

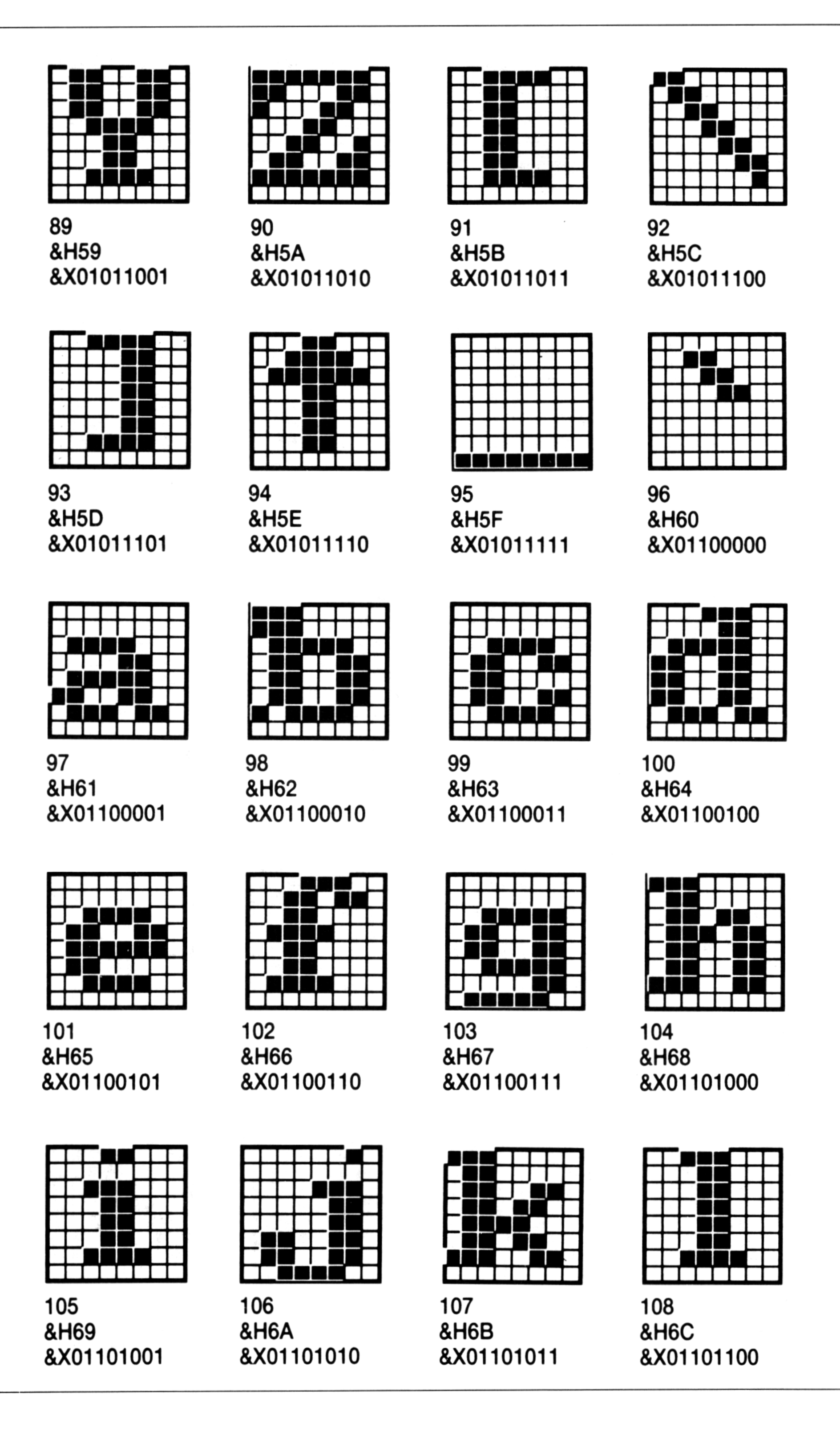
Les caractères représentés ici sont contenus dans la matrice standard 8x8 utilisée pour l'affichage du 6128. L'utilisateur peut générer des caractères et les grouper ou les aligner pour obtenir divers effets spéciaux (voir le paragraphe « Caractères définis par l'utilisateur » du chapitre « A vos heures de loisir... »).

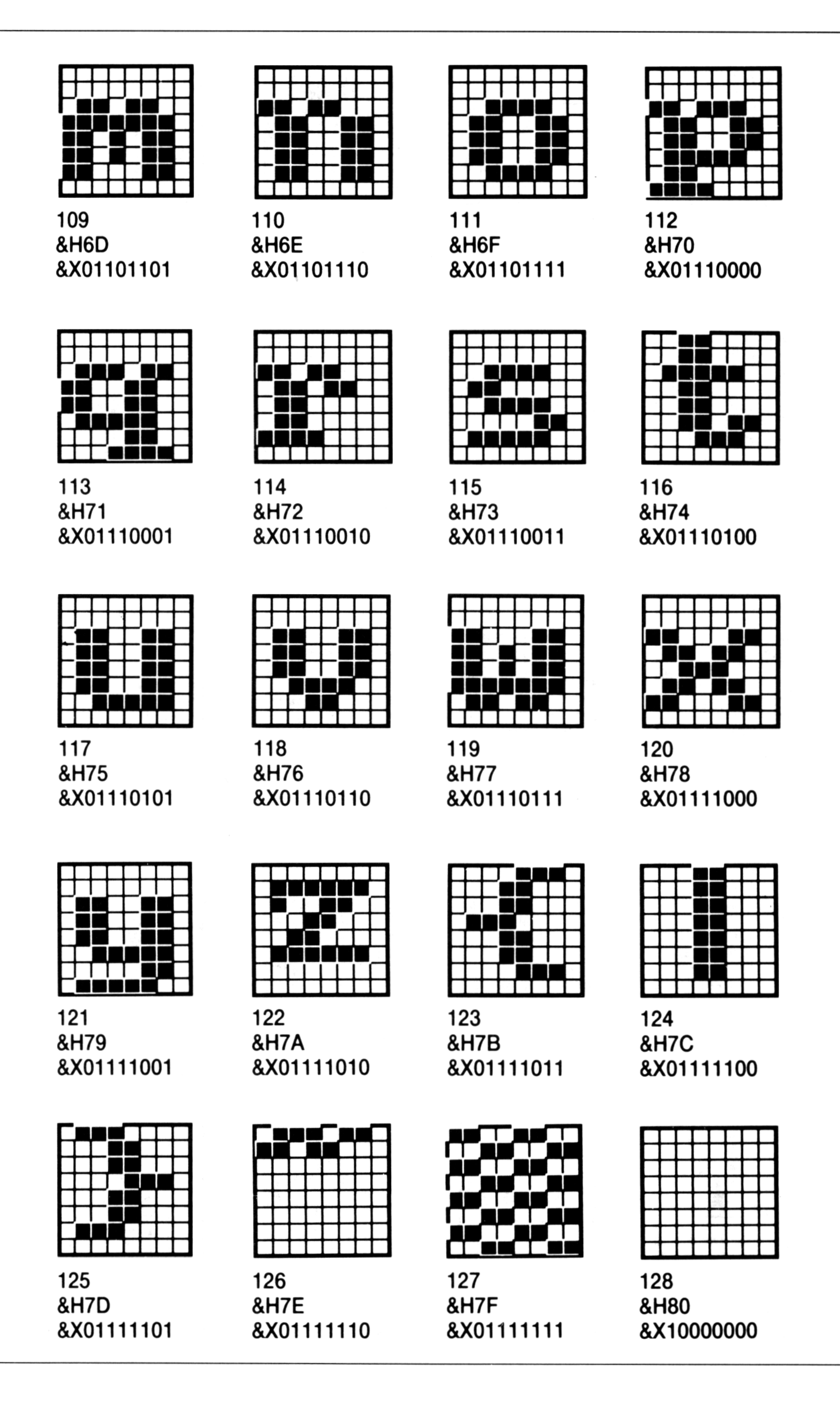


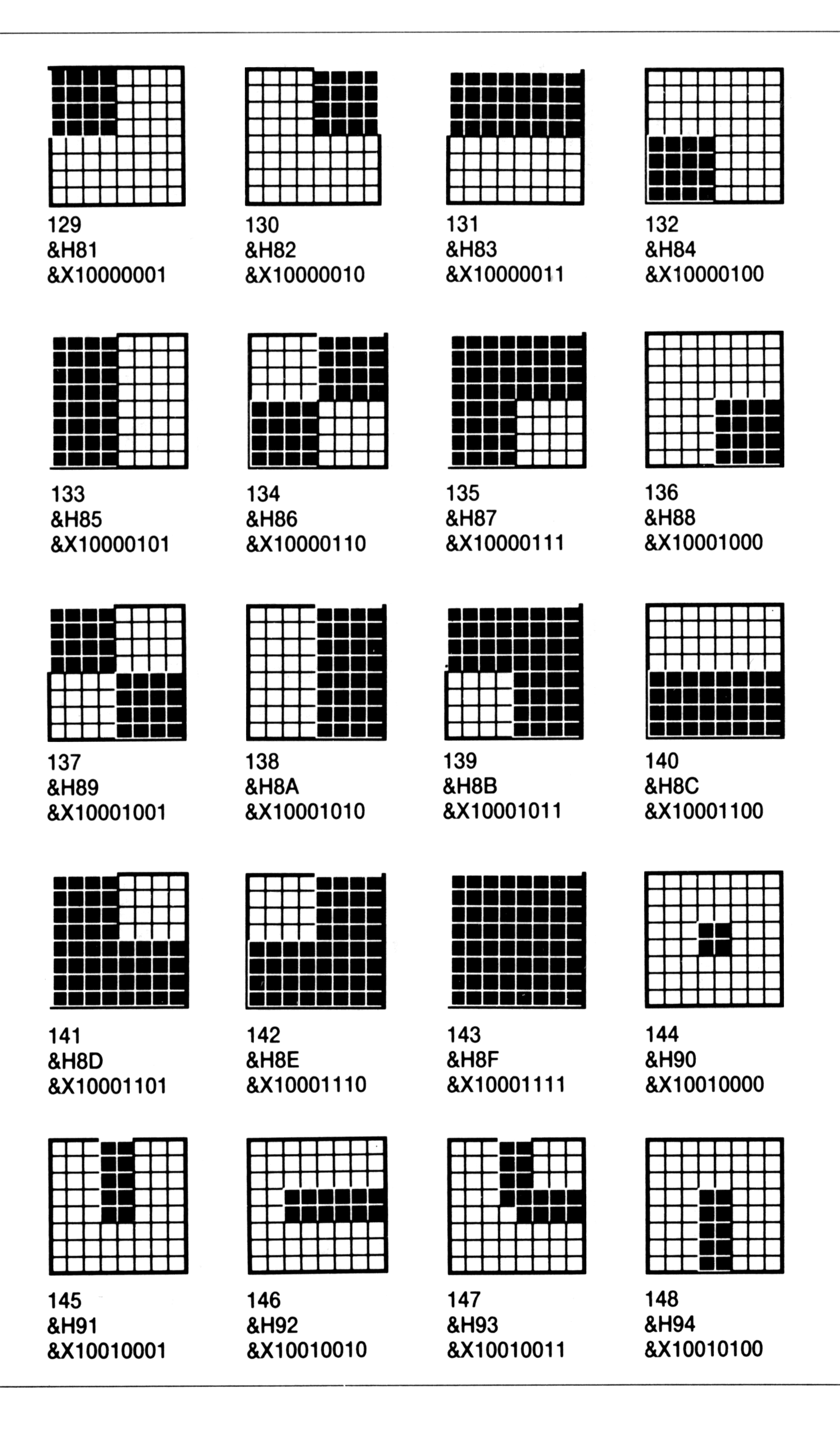

49
&H31
&X00110001
50
&H32
&X00110010
51
&H33
&X00110011
52
&H34
&X00110100
53
&H35
&X00110101
54
&H36
&X00110110
55
&H37
&X00110111
56
&H38
&X00111000
57
&H39
&X00111001
58
&H3A
&X00111010
59
&H3B
&X00111011
60
&H3C
&X00111100
61
&H3D
&X00111101
62
&H3E
&X00111110
63
&H3F
&X00111111
64
&H40
&X01000000
65
&H41
&X01000001
66
&H42
&X01000010
67
&H43
&X01000011
68
&H44
&X01000100

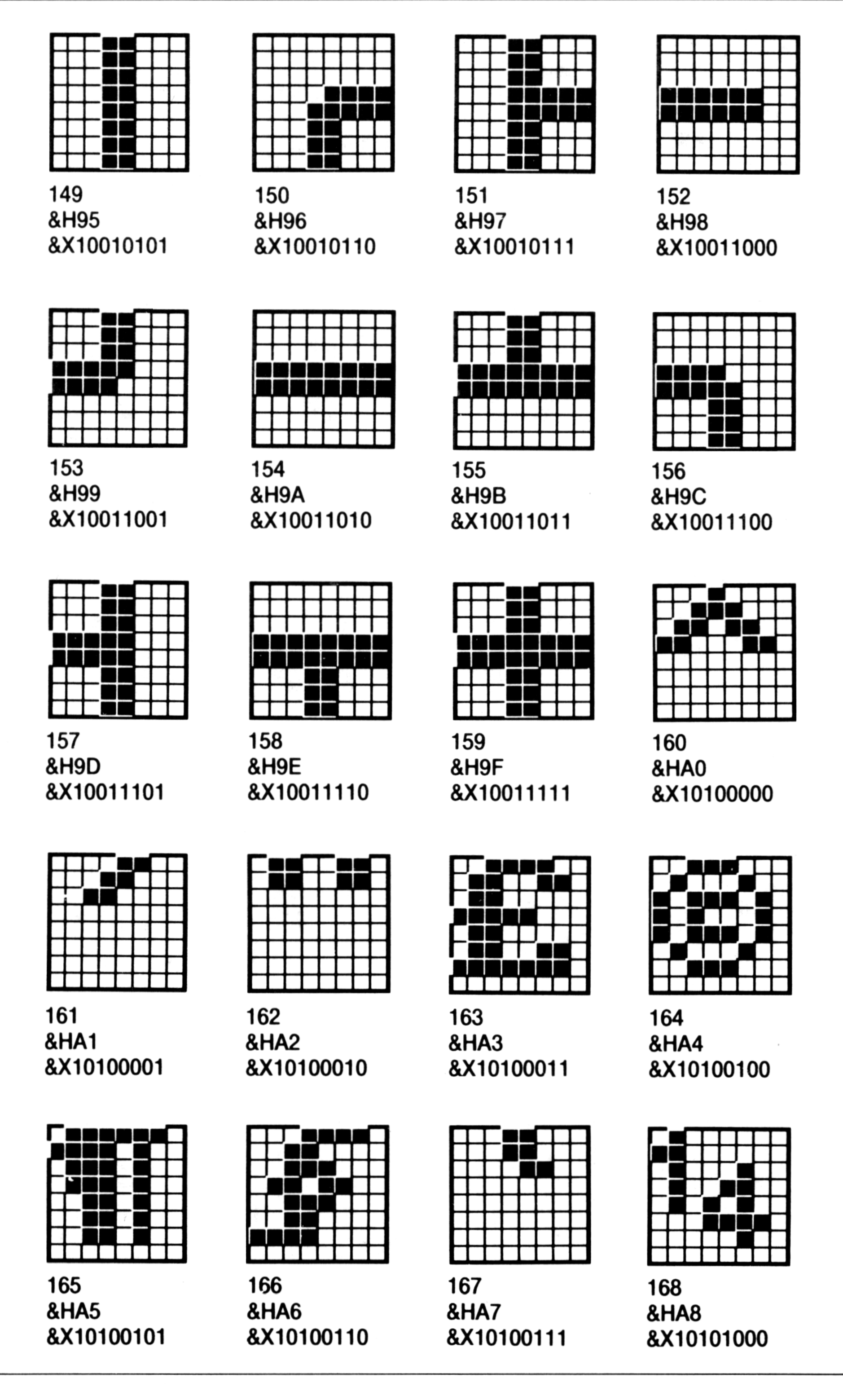


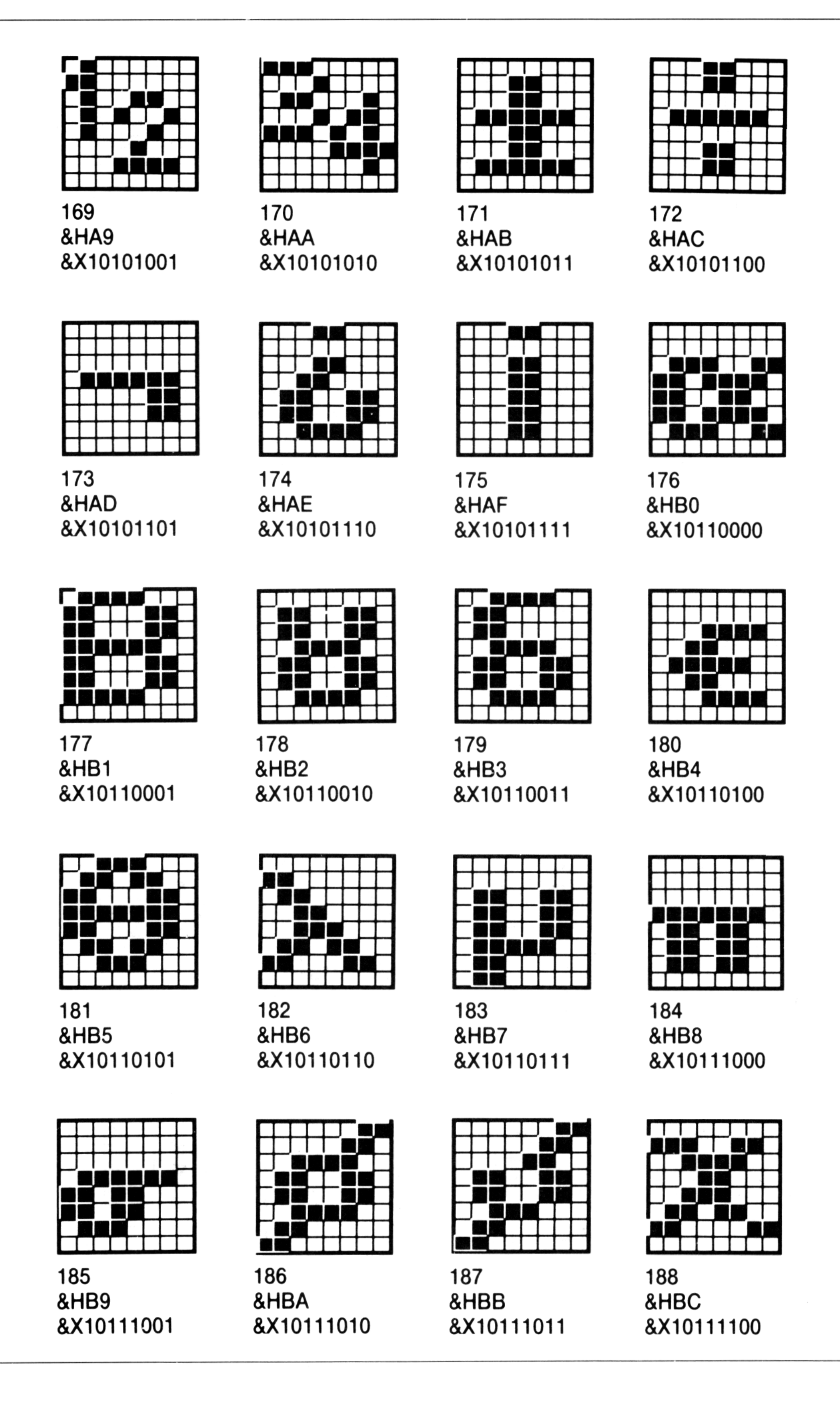
69
&H45
&X01000101
70
&H46
&X01000110
71
&H47
&X01000111
72
&H48
&X01001000
73
&H49
&X01001001
74
&H4A
&X01001010
75
&H4B
&X01001011
76
&H4C
&X01001100
77
&H4D
&X01001101
78
&H4E
&X01001110
79
&H4F
&X01001111
80
&H50
&X01010000
81
&H51
&X01010001
82
&H52
&X01010010
83
&H53
&X01010011
84
&H54
&X01010100
85
&H55
&X01010101
86
&H56
&X01010110
87
&H57
&X01010111
88
&H58
&X01011000

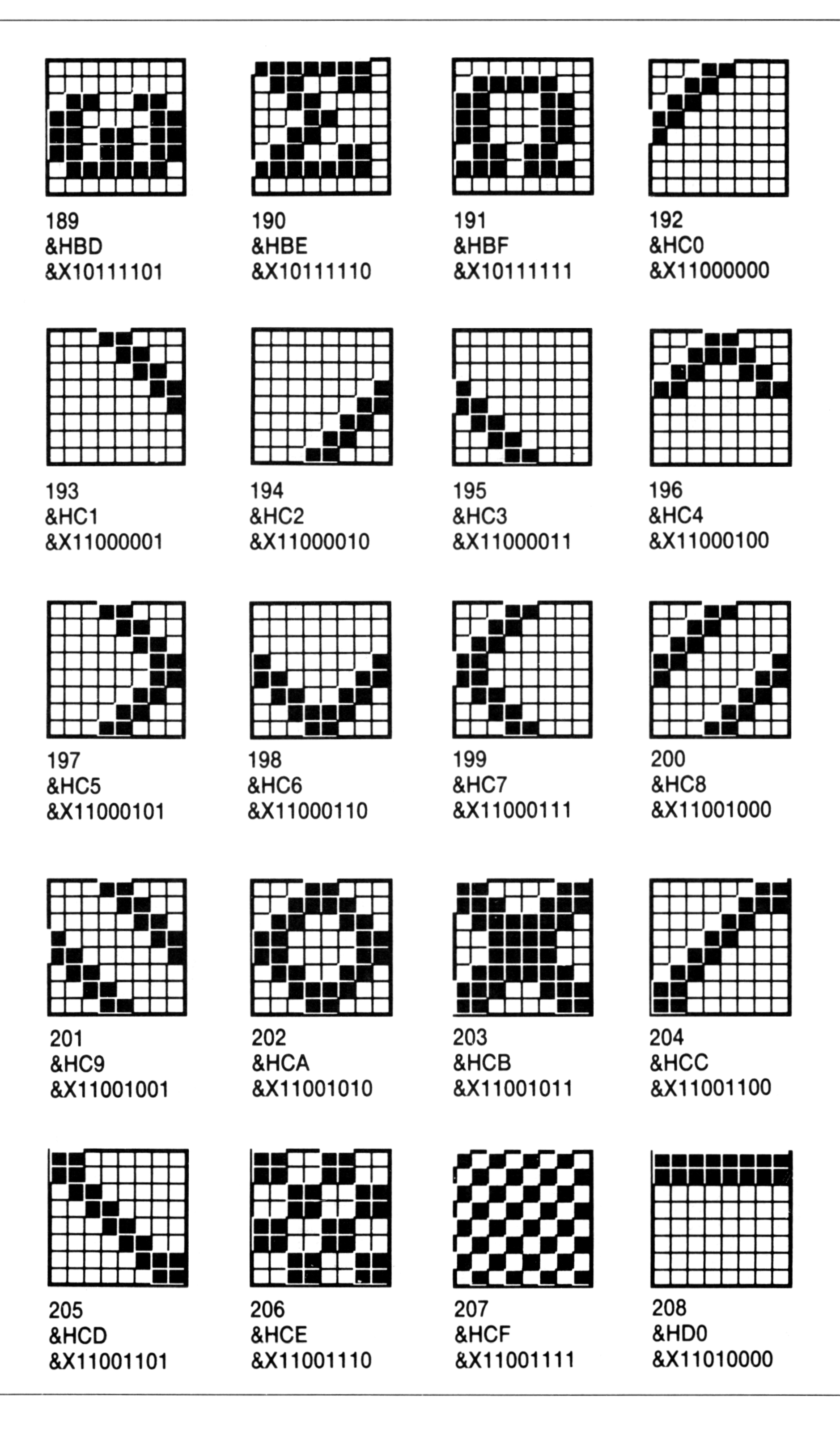

89
&H59
&X01011001
90
&H5A
&X01011010
91
&H5B
&X01011011
92
&H5C
&X01011100
93
&H5D
&X01011101
94
&H5E
&X01011110
95
&H5F
&X01011111
96
&H60
&X01100000
97
&H61
&X01100001
98
&H62
&X01100010
99
&H63
&X01100011
100
&H64
&X01100100
101
&H65
&X01100101
102
&H66
&X01100110
103
&H67
&X01100111
104
&H68
&X01101000
105
&H69
&X01101001
106
&H6A
&X01101010
107
&H6B
&X01101011
108
&H6C
&X01101100

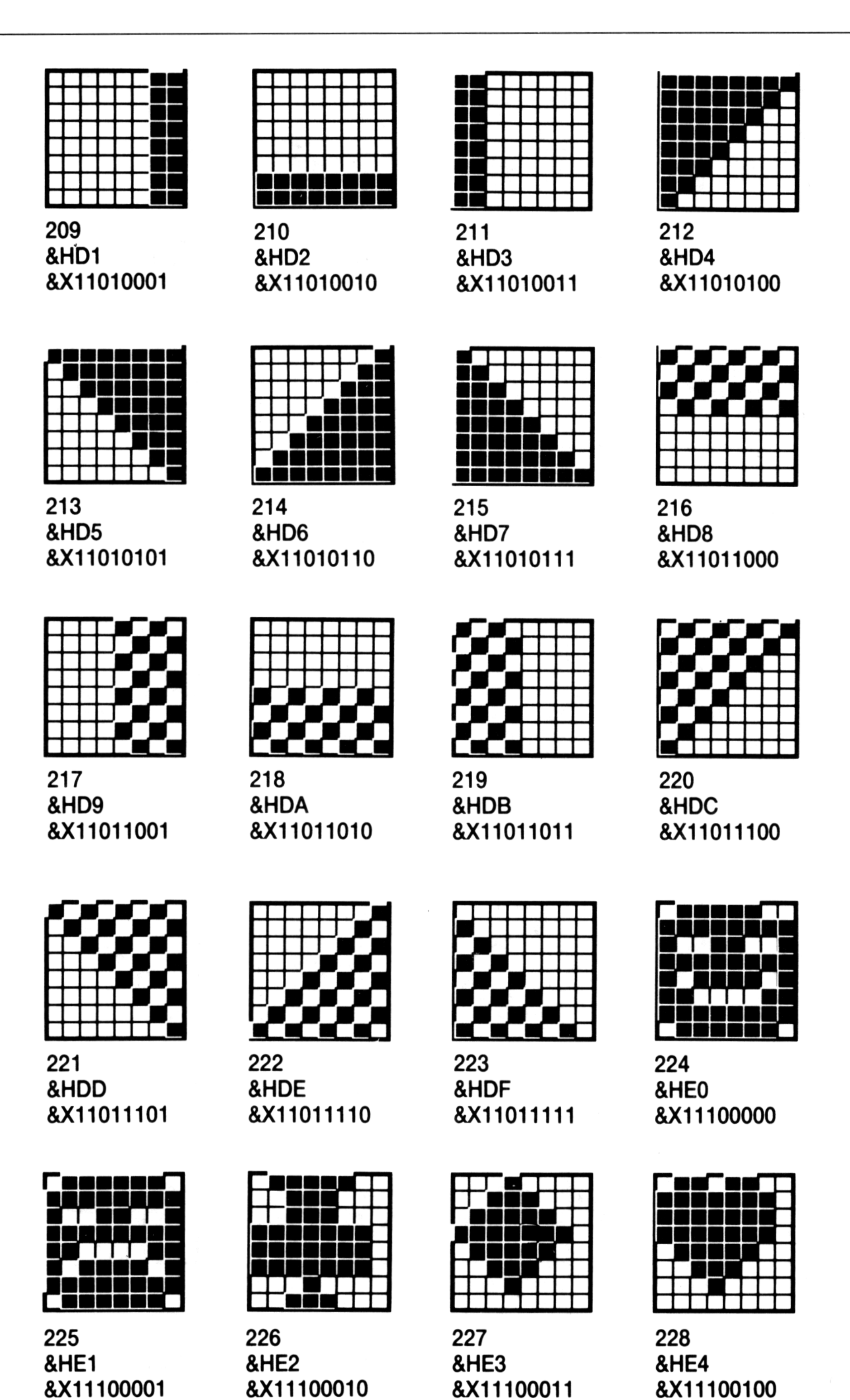

109
&H6D
&X01101101
110
&H6E
&X01101110
111
&H6F
&X01101111
112
&H70
&X01110000
113
&H71
&X01110001
114
&H72
&X01110010
115
&H73
&X01110011
116
&H74
&X01110100
117
&H75
&X01110101
118
&H76
&X01110110
119
&H77
&X01110111
120
&H78
&X01111000
121
&H79
&X01111001
122
&H7A
&X01111010
123
&H7B
&X01111011
124
&H7C
&X01111100
125
&H7D
&X01111101
126
&H7E
&X01111110
127
&H7F
&X01111111
128
&H80
&X10000000


129
&H81
&X10000001
130
&H82
&X10000010
131
&H83
&X10000011
132
&H84
&X10000100
133
&H85
&X10000101
134
&H86
&X10000110
135
&H87
&X10000111
136
&H88
&X10001000
137
&H89
&X10001001
138
&H8A
&X10001010
139
&H8B
&X10001011
140
&H8C
&X10001100
141
&H8D
&X10001101
142
&H8E
&X10001110
143
&H8F
&X10001111
144
&H90
&X10010000
145
&H91
&X10010001
146
&H92
&X10010010
147
&H93
&X10010011
148
&H94
&X10010100

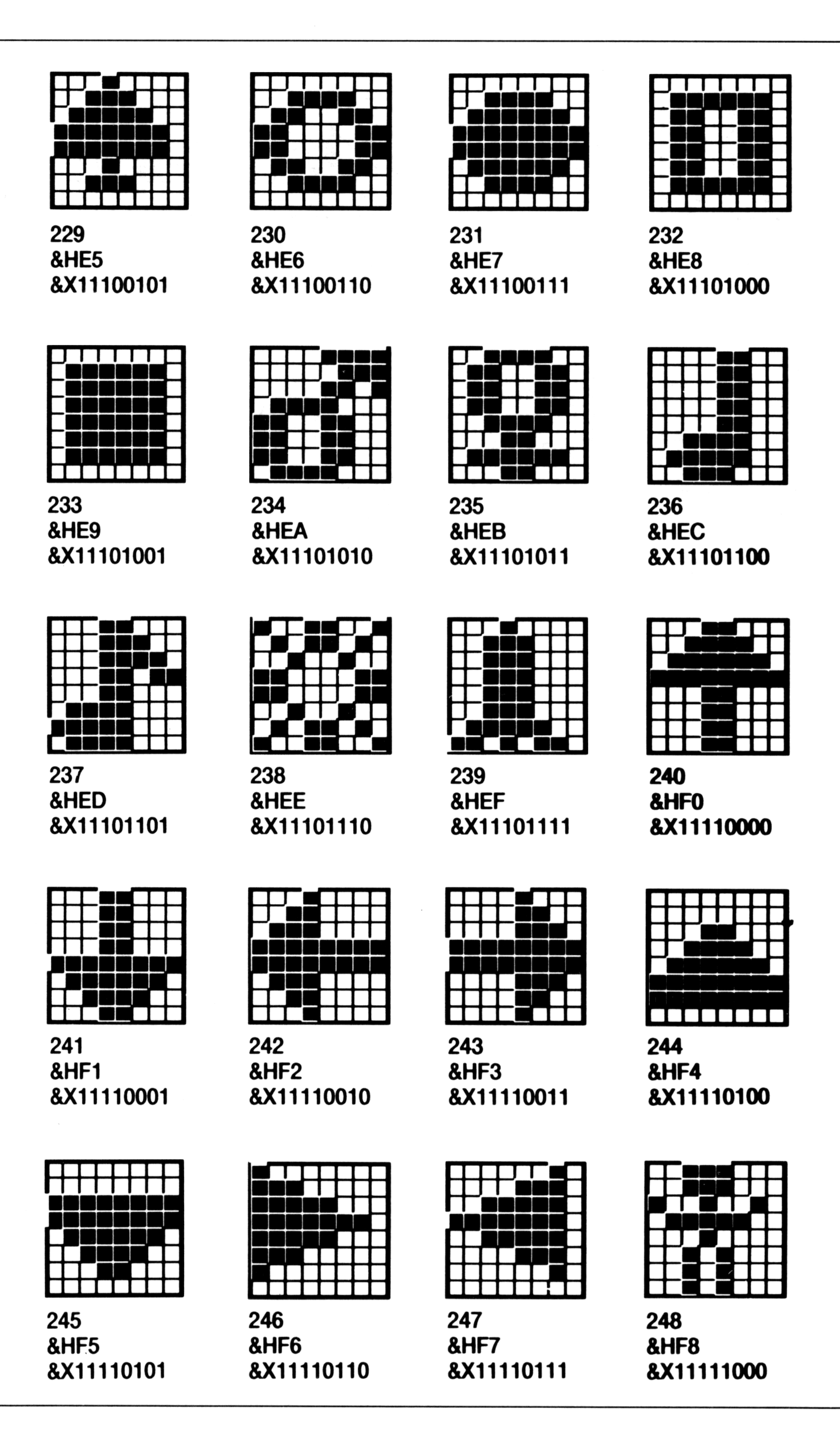

149
&H95
&X10010101
150
&H96
&X10010110
151
&H97
&X10010111
152
&H98
&X10011000
153
&H99
&X10011001
154
&H9A
&X10011010
155
&H9B
&X10011011
156
&H9C
&X10011100
157
&H9D
&X10011101
158
&H9E
&X10011110
159
&H9F
&X10011111
160
&HA0
&X10100000
161
&HA1
&X10100001
162
&HA2
&X10100010
163
&HA3
&X10100011
164
&HA4
&X10100100
165
&HA5
&X10100101
166
&HA6
&X10100110
167
&HA7
&X10100111
168
&HA8
&X10101000

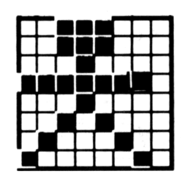

169
&HA9
&X10101001
170
&HAA
&X10101010
171
&HAB
&X10101011
172
&HAC
&X10101100
173
&HAD
&X10101101
174
&HAE
&X10101110
175
&HAF
&X10101111
176
&HB0
&X10110000
177
&HB1
&X10110001
178
&HB2
&X10110010
179
&HB3
&X10110011
180
&HB4
&X10110100
181
&HB5
&X10110101
182
&HB6
&X10110110
183
&HB7
&X10110111
184
&HB8
&X10111000
185
&HB9
&X10111001
186
&HBA
&X10111010
187
&HBB
&X10111011
188
&HBC
&X10111100


189
&HBD
&X10111101
190
&HBE
&X10111110
191
&HBF
&X10111111
192
&HC0
&X11000000
193
&HC1
&X11000001
194
&HC2
&X11000010
195
&HC3
&X11000011
196
&HC4
&X11000100
197
&HC5
&X11000101
198
&HC6
&X11000110
199
&HC7
&X11000111
200
&HC8
&X11001000
201
&HC9
&X11001001
202
&HCA
&X11001010
203
&HCB
&X11001011
204
&HCC
&X11001100
205
&HCD
&X11001101
206
&HCE
&X11001110
207
&HCF
&X11001111
208
&HD0
&X11010000

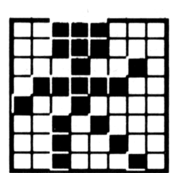


209
&HD1
&X11010001
210
&HD2
&X11010010
211
&HD3
&X11010011
212
&HD4
&X11010100
213
&HD5
&X11010101
214
&HD6
&X11010110
215
&HD7
&X11010111
216
&HD8
&X11011000
217
&HD9
&X11011001
218
&HDA
&X11011010
219
&HDB
&X11011011
220
&HDC
&X11011100
221
&HDD
&X11011101
222
&HDE
&X11011110
223
&HDF
&X11011111
224
&HE0
&X11100000
225
&HE1
&X11100001
226
&HE2
&X11100010
227
&HE3
&X11100011
228
&HE4
&X11100100

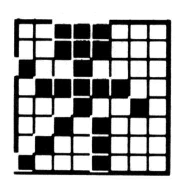

229
&HE5
&X11100101
230
&HE6
&X11100110
231
&HE7
&X11100111
232
&HE8
&X11101000
233
&HE9
&X11101001
234
&HEA
&X11101010
235
&HEB
&X11101011
236
&HEC
&X11101100
237
&HED
&X11101101
238
&HEE
&X11101110
239
&HEF
&X11101111
240
&HF0
&X11110000
241
&HF1
&X11110001
242
&HF2
&X11110010
243
&HF3
&X11110011
244
&HF4
&X11110100
245
&HF5
&X11110101
246
&HF6
&X11110110
247
&HF7
&X11110111
248
&HF8
&X11111000



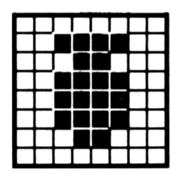
249
&HF9
&X11111001



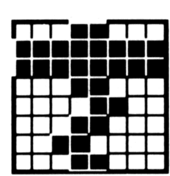
250
&HFA
&X11111010



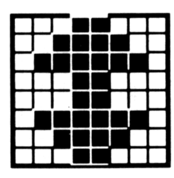
251
&HFB
&X11111011



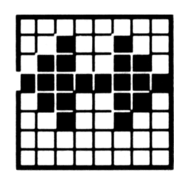
252
&HFC
&X11111100



253
&HFD
&X11111101



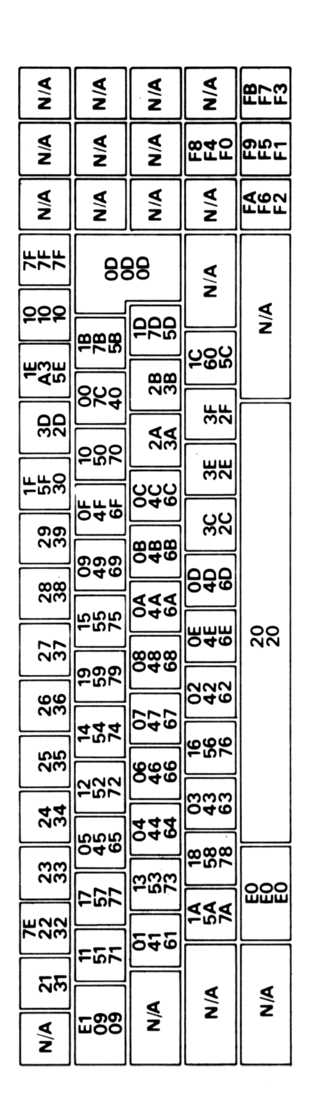
254
&HFE
&X11111110

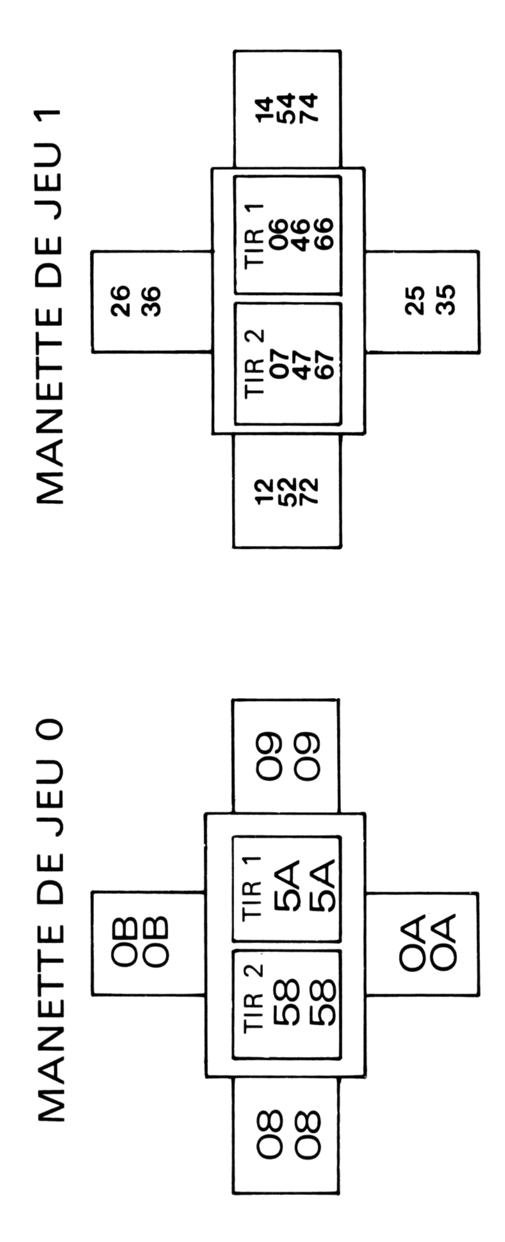


255
&HFF
&X11111111

Partie 4 : Références des touches

Valeurs ASCII par défaut (HEX)





Touches programmables, valeurs et emplacements par défaut

N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A 135 135 135 136 136 136 137 137 137

N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A 132 132 132 133 133 133 134 134 134

N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A 129 129 129 130 130 130 131 131 131

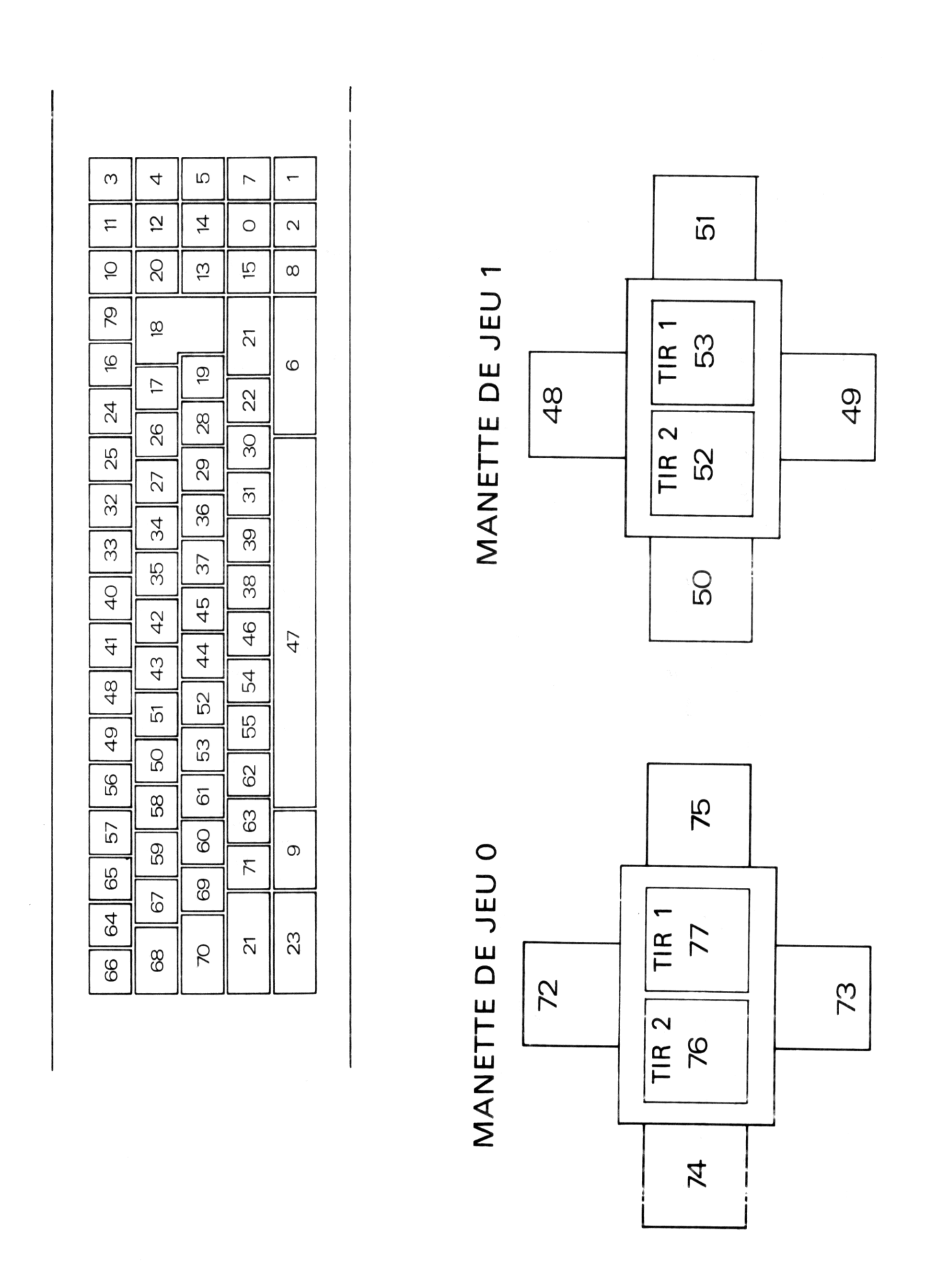
N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A 128 128 128 N/A 138 138 138

N/A N/A N/A 140 139 139 N/A N/A N/A

TOUCHES PROGRAM-MABLES	VALEUR PAR DÉFAUT	
	CARACTÈRE	VALEUR ASCII
0(128)	Ø	&30
1(129)	1	&31
2(130)	2	&32
3(131)	3	&33
4(132)	4	&34
5(133)	5	&35
6(134)	6	&36
7(135)	7	&37
8(136)	8	&38
9(137)	9	&39
10(138)	.	&2E
11(139)	[RETURN]	&0D
12(140)	RUN"[RETURN]	&52 &55 &4E &22 &0D

Remarque : La valeur par défaut des touches programmables 13 à 431 (141 à 159) est égales à 0. Des valeurs et des touches leur sont affectées à l'aide des commandes KEY et KEYDEF respectivement.

Numéros des touches et des positions des manettes de jeu



Partie 5 : Les sons

Notes et périodes sonores

La table suivante donne les périodes sonores pour les notes de la gamme complète de huit octaves.

La fréquence produite n'est pas totalement exacte car elle est calculée à partir d'un nombre entier. L'erreur relative s'obtient en calculant le pourcentage de la différence entre la fréquence théorique et la fréquence réelle par rapport à la fréquence réelle.

NOTE	FRÉQUENCE	PÉRIODE	ERREUR RELATIVE	
C	16.352	3822	-0.007%	
C#	17.324	3608	+0.007%	
D	18.354	3405	-0.007%	
D#	19.445	3214	-0.004%	
E	20.602	3034	+0.009%	
F	21.827	2863	-0.016%	Octave -4
F#	23.125	2703	+0.009%	
G	24.500	2551	-0.002%	
G#	25.957	2408	+0.005%	
A	27,500	2273	+0.012%	
A#	29.135	2145	-0.008%	
B	30.868	2025	+0.011%	

NOTE	FRÉQUENCE	PÉRIODE	ERREUR RELATIVE	
C	32.703	1911	-0.007%	
C#	34.648	1804	+0.007%	
D	36.708	1703	+0.022%	
D#	38.891	1607	-0.004%	
E	41.203	1517	+0.009%	
F	43.654	1432	+0.019%	Octave -3
F#	46.249	1351	-0.028%	
G	48.999	1276	+0.037%	
G#	51.913	1204	+0.005%	
A	55.000	1136	-0.032%	
A#	58.270	1073	+0.039%	
B	61.735	1012	-0.038%	

NOTE	FRÉQUENCE	PÉRIODE	ERREUR RELATIVE	
C	65.406	956	+0.046%	
C#	69.296	902	+0.007%	
D	73.416	851	-0.037%	
D#	77.782	804	+0.058%	
E	82.407	758	-0.057%	
F	87.307	716	+0.019%	**Octave -2**
F#	92.499	676	+0.046%	
G	97.999	638	+0.037%	
G#	103.826	602	+0.005%	
A	110.000	568	-0.032%	
A#	116.541	536	-0.055%	
B	123.471	506	-0.038%	

NOTE	FRÉQUENCE	PÉRIODE	ERREUR RELATIVE	
C	130.813	478	+0.046%	
C#	138.591	451	+0.007%	
D	146.832	426	+0.081%	
D#	155.564	402	+0.058%	
E	164.814	379	-0.057%	
F	174.614	358	+0.019%	**Octave -1**
F#	184.997	338	+0.046%	
G	195.998	319	+0.037%	
G#	207.652	301	+0.005%	
A	220.000	284	-0.032%	
A#	233.082	268	-0.055%	
B	246.942	253	-0.038%	

NOTE	FRÉQUENCE	PÉRIODE	ERREUR RELATIVE	
C	261.626	239	+0.046%	Do médium
C#	277.183	225	-0.215%	
D	293.665	213	+0.081%	
D#	311.127	201	+0.058%	
E	329.628	190	+0.206%	
F	349.228	179	+0.019%	**Octave 0**
F#	369.994	169	+0.046%	
G	391.995	159	-0.277%	
G#	415.305	150	-0.328%	
A	440.000	142	-0.032%	A international
A#	466.164	134	-0.055%	
B	493.883	127	+0.356%	

NOTE	FRÉQUENCE	PÉRIODE	ERREUR RELATIVE	
C	523.251	119	-0.374%	
C#	554.365	113	+0.229%	
D	587.330	106	-0.390%	
D#	622.254	100	-0.441%	
E	659.255	95	+0.206%	
F	698.457	89	-0.543%	**Octave 1**
F#	739.989	84	-0.548%	
G	783.991	80	+0.350%	
G#	830.609	75	-0.328%	
A	880.000	71	-0.032%	
A#	932.328	67	-0.055%	
B	987.767	63	-0.435%	

NOTE	FRÉQUENCE	PÉRIODE	ERREUR RELATIVE	
C	1046.502	60	+0.462%	
C#	1108.731	56	-0.662%	
D	1174.659	53	-0.390%	
D#	1244.508	50	-0.441%	
E	1318.510	47	-0.855%	
F	1396.913	45	+0.574%	
F#	1479.978	42	-0.548%	**Octave 2**
G	1567.982	40	+0.350%	
G#	1661.219	38	+0.992%	
A	1760.000	36	+1.357%	
A#	1864.655	34	+1.417%	
B	1975.533	32	+1.134%	

NOTE	FRÉQUENCE	PÉRIODE	ERREUR RELATIVE	
C	2093.004	30	+0.462%	
C#	2217.461	28	-0.662%	
D	2349.318	27	+1.469%	
D#	2489.016	25	-0.441%	
E	2637.021	24	+1.246%	
F	2793.826	22	-1.685%	
F#	2959.955	21	-0.548%	**Octave 3**
G	3135.963	20	+0.350%	
G#	3322.438	19	+0.992%	
A	3520.000	18	+1.357%	
A#	3729.310	17	+1.417%	
B	3951.066	16	+1.134%	

Ces valeurs sont toutes calculées à partir du LA International comme suit :

```
FREQUENCE =    440*(2↑(OCTAVE+((N-10)/12)))
PERIODE =      ROUND(125000/FREQUENCE)
```

où **N** est égal à **1** pour do, **2** pour **do dièse**, **3** pour **ré**, etc.

Partie 6 : Messages d'erreur du BASIC

```
1 Unexpected NEXT    (NEXT inattendu)
```

Une commande **NEXT** a été rencontrée sans que la commande **FOR** n'ait lancé de boucle ou bien la variable suivant **NEXT** ne correspond pas à celle de la boucle **FOR**.

```
2 Syntax Error    (erreur de syntaxe)
```

Le BASIC ne comprend pas la ligne à cause d'une construction non permise (très souvent une faute de frappe).

```
3 Unexpected RETURN  (RETURN inattendu)
```

Une commande **RETURN** survient sans qu'il y ait de sous-programme en cours.

```
4 DATA exhausted     (il n'y a plus de donnees (DATA))
```

Une commande **READ** a essayé de lire une ligne de **DATA** épuisée.

```
5 Improper argument  (argument incorrect)
```

Erreur d'ordre général. L'argument d'une fonction ou le paramètre d'une commande n'est pas acceptable.

```
6 Overflow  (depassement arithmetique)
```

Se produit lorsqu'une opération arithmétique dépasse les limites. Le chiffre en virgule flottante est devenu trop grand (supérieur à 1.7E↑38) ou a fait l'objet d'une tentative de conversion en un nombre entier trop grand.

```
7 Memory full (memoire saturee)
```

Le programme ou ses variables sont trop grands pour la mémoire ou la structure des boucles est trop compliquée (trop de **GOSUB**, **WHILE** et **FOR** imbriqués).

A chaque fichier ouvert est affecté un tampon de mémoire, ce qui peut être une cause de limitation pour la commande **MEMORY**.

```
8 Line does not exist    (ligne inexistante)
```

Le numéro de ligne référencé n'existe pas en mémoire.

```
9 Subscript out of range (indice hors limite)
```

Un des indices de votre tableau est trop grand ou trop petit.

```
10 Array already dimensioned (tableau deja dimensionne)
```

Un des tableaux d'une instruction **DIM** a déjà été défini.

```
11 Division by zero   (division par zero)
```

L'ordinateur n'aime pas diviser par zéro, que ce soit réel, entier, etc...

```
12 Invalid direct command  (commande directe non valable)
```

La commande n'est pas acceptable en mode direct.

```
13 Type mismatch  (types de variable ne correspondant pas)
```

On a donné une valeur numérique pour une chaîne alphanumérique ou vice-versa ou un nombre non valable a été découvert par une commande **READ** ou **INPUT**.

```
14 String space full  (espace reserve aux chaines sature)
```

Il y a tellement de chaînes qu'il n'y a plus de place, même après une remise en ordre.

```
15 String too long  (chaine trop longue)
```

Une chaîne a plus de 255 caractères, ce qui peut arriver lors d'une concaténation de chaînes.

```
16 String expression too complex (chaine trop compliquee)
```

Des expressions de chaînes peuvent produire des valeurs intermédiaires qui, si elles sont trop nombreuses, conduisent le BASIC à donner ce message.

```
17 Cannot CONTinue (on ne peut pas CONTinuer)
```

Le programme ne peut pas poursuivre son exécution avec **CONT**, qui sert après une commande **STOP**, **[ESC][ESC]** ou une erreur. Notez que si le programme a été modifié entre-temps, il est impossible de le relancer par cette commande.

```
18 Unknown user function (fonction inconnue au bataillon)
```

On a oublié de définir la fonction **FN** avec la commande **DEF FN** auparavant.

```
19 RESUME missing  (commande RESUME absente)
```

On a trouvé la fin de programme alors que celui-ci procédait à un traitement d'erreur (à la suite d'une déclaration **ON ERROR GOTO**).

```
20 Unexpected RESUME  (RESUME inattendu)
```

On tombe sur une commande **RESUME** sans être dans un sous-programme de type **ON ERROR GOTO**.

```
21 Direct command found (commande directe tombant du ciel!)
```

En chargeant un programme, une ligne sans numéro s'est présentée.

```
22 Operand missing  (operande absent)
```

Le BASIC vient de tomber sur une expression incomplète.

```
23 Line too long (ligne trop longue)
```

Le BASIC n'accepte pas les lignes de plus de 255 caractères.

```
24 EOF met (rencontre d'une fin de fichier)
```

EOF = End Of File = fin de fichier ; le programme a effectué une tentative de lecture après une fin de fichier.

```
25 File type error (erreur dans le type de fichier)
```

Le fichier n'est pas du type requis. **OPENIN** peut seulement ouvrir des fichiers de texte ASCII. **LOAD**, **RUN**, etc., ne fonctionnent qu'avec des fichiers produits par **SAVE**.

26 NEXT missing (NEXT manquant)

On ne peut pas trouver le **NEXT** qui corresponde à une commande **FOR**.

27 File already open (fichier deja ouvert)

Une commande **OPENIN** ou **OPENOUT** est exécutée avant que le fichier déjà ouvert n'ait été fermé.

28 Unknown command (commande inconnue)

Le BASIC ne trouve pas de références à cette commande externe.

29 WEND missing (WEND manquant)

La boucle commencée par **WHILE** n'est pas terminée par **WEND**.

30 Unexpected WEND (wend inattendu)

Un **WEND** est découvert en dehors d'une boucle **WHILE**, ou un **WEND** ne correspond pas au **WHILE** de la boucle.

31 File not open (fichier non ouvert)

(Voir paragraphe ci-dessous, « Erreurs de disquette ».)

32 Broken in (interrompu)

(Voir paragraphe ci-dessous, « Erreurs de disquette ».)

Erreurs sur disquette en AMSDOS

Plusieurs erreurs peuvent se produire lors du traitement des opérations d'archivage. Bien que le BASIC les regroupe sous le numéro d'**ERR**eur **32**, vous pouvez obtenir de plus amples informations en appelant la fonction **DERR**. Voici la signification des valeurs qu'elle renvoie :

Erreur AMSDOS	Valeur DERR	Source de l'erreur
0	**Ø** ou **22**	Activation de **[ESC]**.
14	**142** (128 + 14)	Etat du canal non valable.
15	**143** (128 + 15)	Fin de fichier matérielle.
16	**144** (128 + 16)	Mauvaise commande, généralement nom de fichier incorrect.
17	**145** (128 + 17)	Fichier déjà existant.
18	**146** (128 + 18)	Fichier non existant.
19	**147** (128 + 19)	Catalogue saturé.
20	**148** (128 + 20)	Disquette pleine.
21	**149** (128 + 21)	Changement de la disquette avec fichiers ouverts.
22	**15Ø** (128 + 22)	Fichier en lecture seulement.
26	**154** (128 + 26)	Fin de fichier logicielle.

Si AMSDOS a déjà rapporté une erreur, le bit 7 a pris la valeur 1, décalant celle de **DERR** de 128.

Les autres valeurs rapportées par **DERR** proviennent du contrôleur de la disquette, le bit 6 étant toujours sur 1. Le bit 7 indique si AMSDOS a rapporté l'erreur ou non (voir ci-dessus). Voici la signification de chacun des bits :

Bit	Signification
0	Adresse manquante.
1	Ecriture impossible. Disquette protégée.
2	Pas de donnée. Secteur introuvable.
3	Unité non prête. Pas de disquette dans l'unité.
4	Surcharge.
5	Erreur de donnée. Erreur CRC (cyclic redundancy check).
6	Toujours à 1 pour indiquer une erreur venant du contrôleur de disquette.
7	Sur 1 si AMSDOS a déjà rapporté l'erreur.

ERR peut envoyer **31** si vous avez tenté l'accès à une disquette sans aucun fichier ouvert. Une procédure classique d'utilisation de **ERR** et **DERR** consiste à inclure l'instruction **ON ERROR GOTO** afin d'appeler un sous-programme déterminant la valeur (**31 ou 32**) de ERR puis, en cas d'erreur **32**, utilisant l'indicateur **DERR** pour afficher la nature exacte de l'erreur. Par exemple :

```
10 ON ERROR GOTO 1000
20 OPENOUT "monfich.asc"
30 WRITE #9,"donnee-test"
40 CLOSEOUT
50 END
1000 amsdoserr=(DERR AND &7F):REM bit 7 force a 0
1010 IF ERR<31 THEN END
1020 IF ERR=31 THEN PRINT "etes-vous sur d'avoir tape la lig
ne 20 ?":END
1030 IF amsdoserr=20 THEN PRINT "Il n'y a plus de place sur
la disquette ":END
1040 IF amsdoserr=&X1000010 THEN PRINT"placez une disquette
non protegee dans  l'unite, puis appuyez sur n'importe     q
uelle touche ":WHILE INKEY$="":WEND:RESUME
1050 END
```

Partie 7 : Mots clés du BASIC (également appelés mots réservés)

Voici une liste des mots clés du BASIC Schneider 6128. Ils sont réservés et ne peuvent être utilisés comme variables.

```
ABS, AFTER, AND, ASC, ATN, AUTO

BIN$, BORDER

CALL, CAT, CHAIN, CHR$, CINT, CLEAR, CLG, CLOSEIN,
CLOSEOUT, CLS, CONT, COPYCHR$, COS, CREAL, CURSOR

DATA, DEC$, DEF, DEFINT, DEFREAL, DEFSTR, DEG, DELETE,
DERR, DI, DIM, DRAW, DRAWR

EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE, ERL, ERR,
ERROR, EVERY, EXP

FILL, FIX, FN, FOR, FRAME, FRE
```

GOSUB, GOTO, GRAPHICS

HEX$, HIMEM

IF, INK, INKEY, INKEY$, INP, INPUT, INSTR, INT

JOY

KEY

LEFT$, LEN, LET, LINE, LIST, LOAD, LOCATE, LOG, LOG10
LOWER$

MASK, MAX, MEMORY, MERGE, MID$, MIN, MOD, MODE, MOVE,
MOVER

NEXT, NEW, NOT

ON, ON BREAK, ON ERROR GOTO 0, ON SQ, OPENIN, OPENOUT,
OR, ORIGIN, OUT

PAPER, PEEK, PEN, PI, PLOT, PLOTR, POKE, POS, PRINT

RAD, RANDOMIZE, READ, RELEASE, REM, REMAIN, RENUM,
RESTORE, RESUME, RETURN, RIGHT$, RND, ROUND, RUN

SAVE, SGN, SIN, SOUND, SPACE$, SPC, SPEED, SQ, SQR,
STEP, STOP, STR$, STRING$, SWAP, SYMBOL

TAB, TAG, TAGOFF, TAN, TEST, TESTR, THEN, TIME, TO,
TROFF, TRON

UNT, UPPER$, USING

VAL, VPOS

WAIT, WEND, WHILE, WIDTH, WINDOW, WRITE

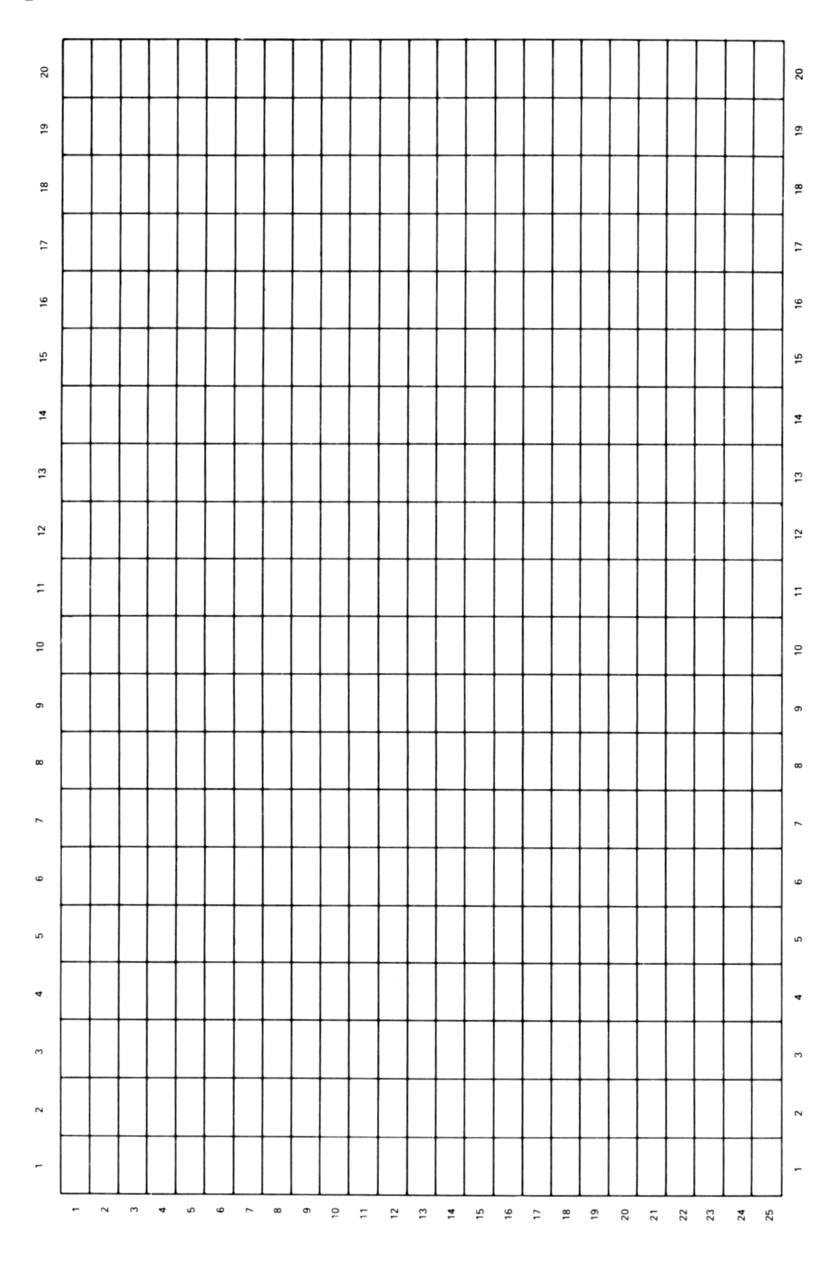
XOR, XPOS

YPOS

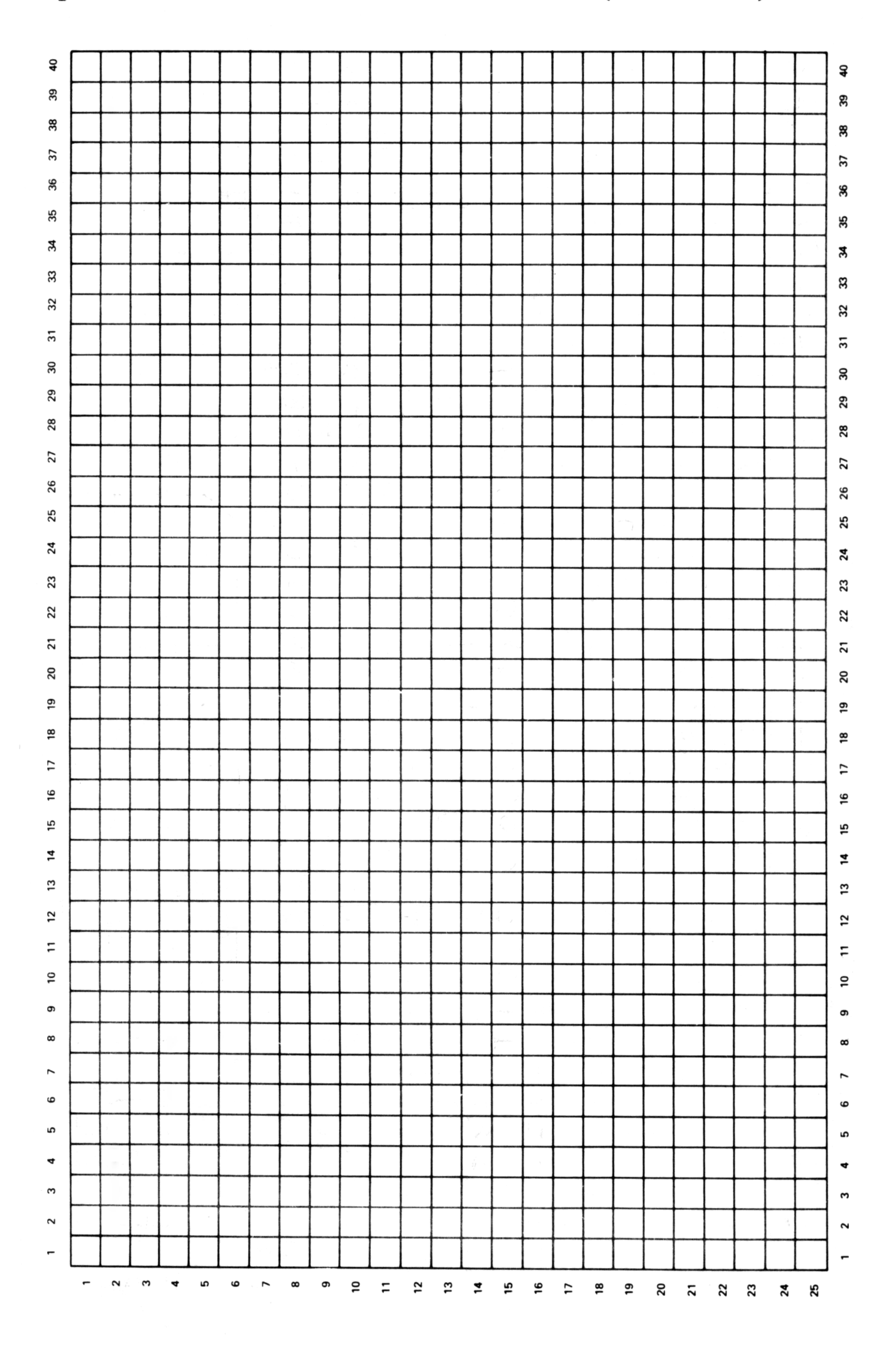
ZONE

Partie 8 : Grilles

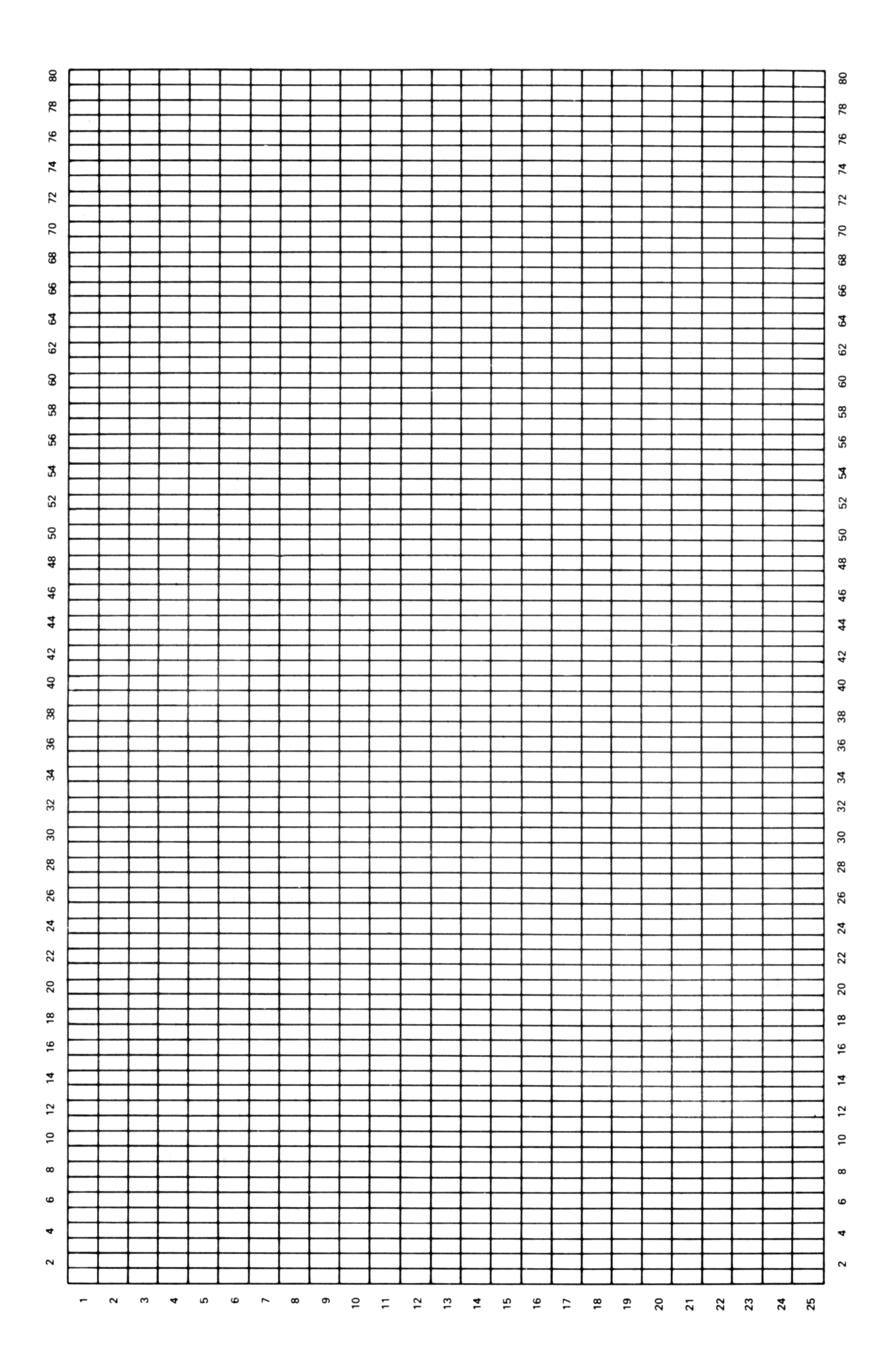
Grille pour textes et fenêtres d'écran - MODE 0 (20 colonnes)



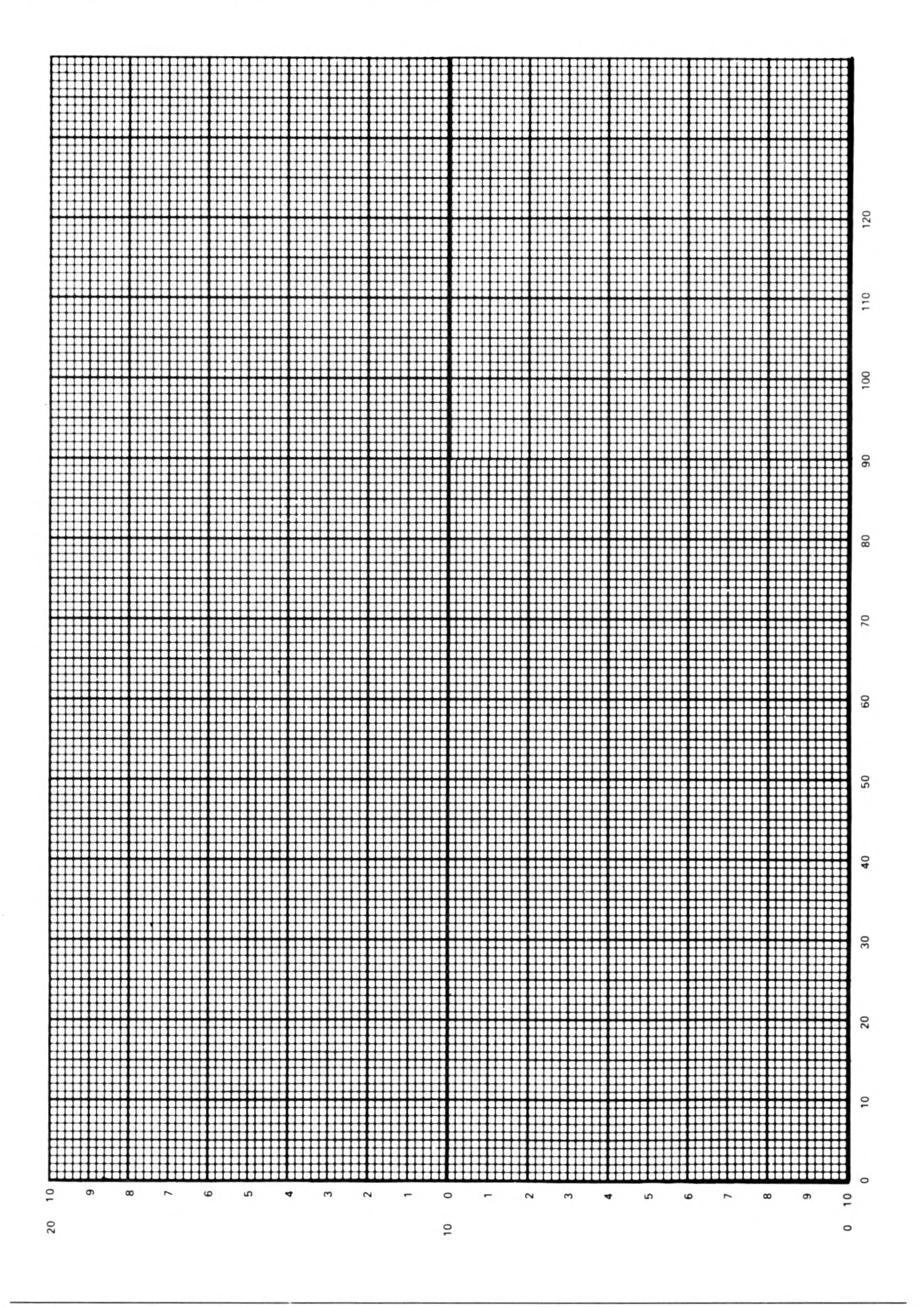
Grille pour textes et fenêtres d'écran - MODE 1 (40 colonnes)



Grille pour textes et fenêtres d'écran - MODE 2 (80 colones)

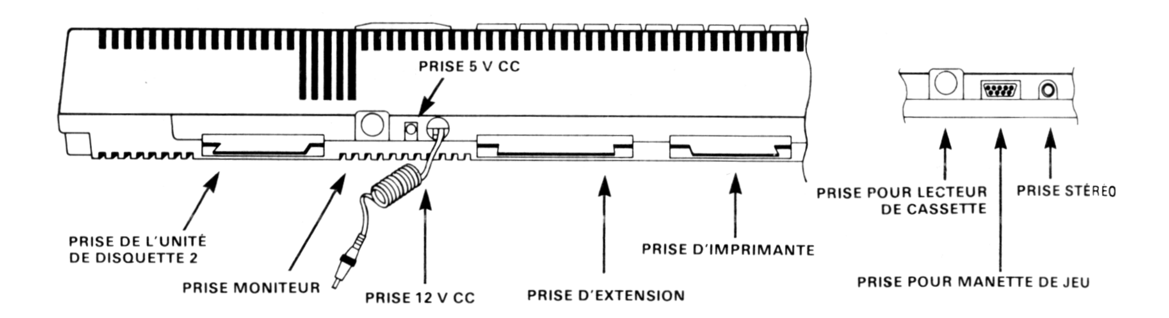


Enveloppe des sons/Grille de musique

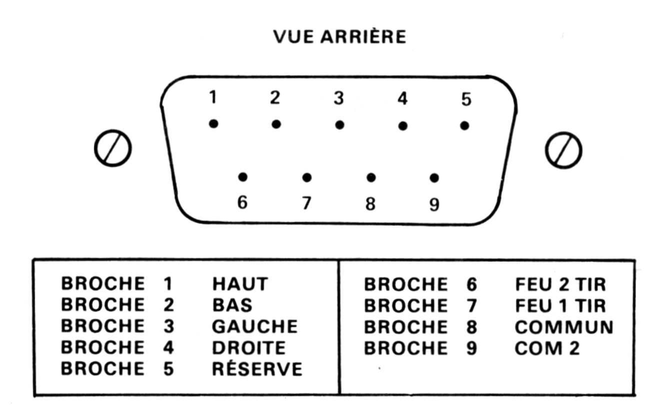


Partie 9 : Connexions

Entrées/Sorties du 6128

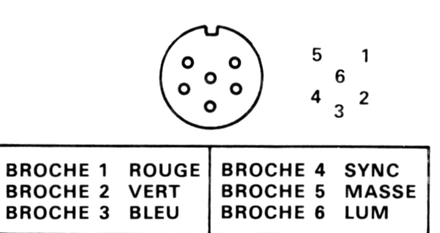


Prise de manette de jeu



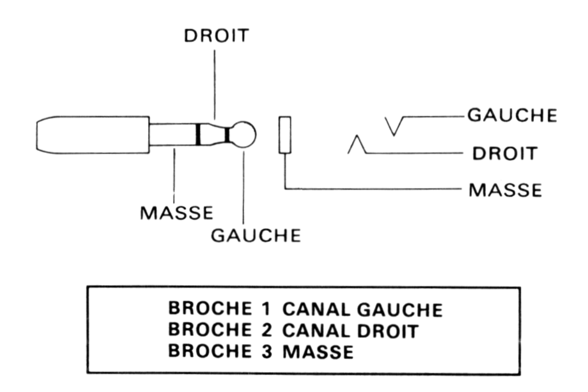
BROCHE 1	HAUT	BROCHE 6	FEU 2 TIR
BROCHE 2	BAS	BROCHE 7	FEU 1 TIR
BROCHE 3	GAUCHE	BROCHE 8	COMMUN
BROCHE 4	DROITE	BROCHE 9	COM 2
BROCHE 5	RÉSERVE		

Prise moniteur



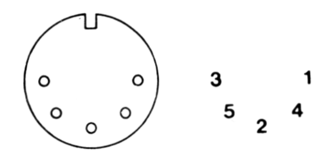
BROCHE 1 ROUGE	BROCHE 4 SYNC
BROCHE 2 VERT	BROCHE 5 MASSE
BROCHE 3 BLEU	BROCHE 6 LUM

Prise stéréo



BROCHE 1 CANAL GAUCHE
BROCHE 2 CANAL DROIT
BROCHE 3 MASSE

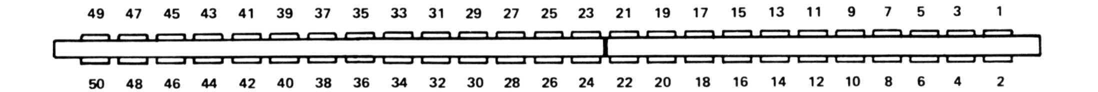
Prise de lecteur de cassette



BROCHE 1 TELECOMMANCE	BROCHE 4 DONNEES ENTRANTES
BROCHE 2 MASSE	BROCHE 5 DONNEES SORTANTES
BROCHE 3 TELECOMMANDE	

Prise d'extension

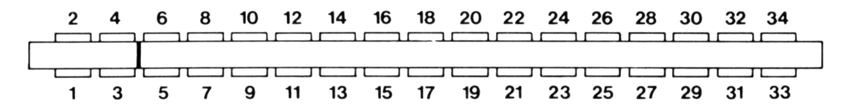
VUE ARRIÈRE



PIN 1	SOUND	PIN 18	A0	PIN 35	INT
PIN 2	GND	PIN 19	D7	PIN 36	NMI
PIN 3	A15	PIN 20	D6	PIN 37	BUSR2
PIN 4	A14	PIN 21	D5	PIN 38	BUSAK
PIN 5	A13	PIN 22	D4	PIN 39	READY
PIN 6	A12	PIN 23	D3	PIN 40	BUS RESET
PIN 7	A11	PIN 24	D2	PIN 41	RESET
PIN 8	A10	PIN 25	D1	PIN 42	ROMEN
PIN 9	A9	PIN 26	D0	PIN 43	ROMDIS
PIN 10	A8	PIN 27	+ 5v	PIN 44	RAMRD
PIN 11	A7	PIN 28	MREQ	PIN 45	RAMDIS
PIN 12	A6	PIN 29	M1	PIN 46	CURSOR
PIN 13	A5	PIN 30	RFSH	PIN 47	L. PEN
PIN 14	A4	PIN 31	IORQ	PIN 48	EXP
PIN 15	A3	PIN 32	RD	PIN 49	GND
PIN 16	A2	PIN 33	WR	PIN 50	φ
PIN 17	A1	PIN 34	HALT		

Prise de l'unité de disquette 2

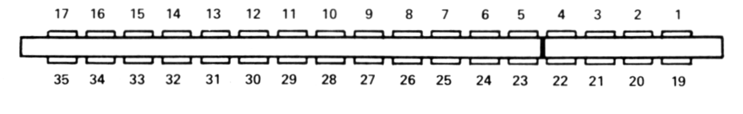
VUE ARRIÈRE



PIN 1	READY	PIN 18	GND
PIN 2	GND	PIN 19	MOTOR ON
PIN 3	SIDE 1 SELECT	PIN 20	GND
PIN 4	GND	PIN 21	N/C
PIN 5	READ DATA	PIN 22	GND
PIN 6	GND	PIN 23	DRIVE SELECT 1
PIN 7	WRITE PROTECT	PIN 24	GND
PIN 8	GND	PIN 25	N/C
PIN 9	TRACK 0	PIN 26	GND
PIN 10	GND	PIN 27	INDEX
PIN 11	WRITE GATE	PIN 28	GND
PIN 12	GND	PIN 29	N/C
PIN 13	WRITE DATA	PIN 30	GND
PIN 14	GND	PIN 31	N/C
PIN 15	STEP	PIN 32	GND
PIN 16	GND	PIN 33	N/C
PIN 17	DIRECTION SELECT	PIN 34	GND

Porte pour imprimante

VUE ARRIÈRE



PIN 1	$\overline{\text{STROBE}}$	PIN 19	GND
PIN 2	D0	PIN 20	GND
PIN 3	D1	PIN 21	GND
PIN 4	D2	PIN 22	GND
PIN 5	D3	PIN 23	GND
PIN 6	D4	PIN 24	GND
PIN 7	D5	PIN 25	GND
PIN 8	D6	PIN 26	GND
PIN 9	GND	PIN 27	GND
PIN 11	BUSY	PIN 28	GND
PIN 14	GND	PIN 33	GND
PIN 16	GND	Autres broches : NC	

Partie 10 : Imprimante

Interfaçage

Le 6128 peut être connecté à une imprimante standard compatible Centronics.

Le câble de l'imprimante est une simple connexion entre la prise PRINTER située à l'arrière de l'ordinateur et le connecteur de l'imprimante parallèle. Le circuit imprimé de l'ordinateur présente deux contacts de moins que le connecteur de l'imprimante. Ceci permet l'utilisation d'un connecteur standard pour circuit imprimé.

La partie 9 de ce chapitre illustre en détail la configuration des broches.

Le câble doit permettre la connexion de la broche 1 de l'ordinateur à la broche 1 de l'imprimante ; de la broche 19 de l'ordinateur à la broche 19 de l'imprimante, etc. Toutefois, les broches 18 et 36 de l'imprimante de doivent pas être connectées à l'ordinateur.

Bien que la rangée supérieure de la prise PRINTER comporte 17 pattes, le numéro de la première patte de la rangée inférieure est 19 (et non pas 18). De cette manière, chaque broche de l'imprimante se trouve reliée à la piste du connecteur plat de l'ordinateur portant exactement le même numéro qu'elle.

La synchronisation de l'ordinateur à l'imprimante s'effectue par l'intermédiaire du signal BUSY (broche 11). Si l'imprimante n'est pas en fonction, l'ordinateur reste en attente.

On obtient la sortie sur l'imprimante en envoyant les données sur le canal # 8.

Le port d'impression du 6128 est conçu pour recevoir un modèle d'imprimante matricielle bon marché. Nanti d'une interface parallèle standard, il vous sera toutefois possible de doter votre ordinateur d'une imprimante à marguerite, d'une table traçante ou d'une imprimante polychrome à jet d'encre.

L'imprimante Schneider DMP dispose d'un programme spécialisé facilitant l'impression graphique comme la recopie d'écran.

Configuration de l'imprimante

Un dispositif permet l'impression des caractères spéciaux affichables à l'écran et acceptés par l'imprimante, en dépit de la différence du code utilisé dans l'un et l'autre cas. La plupart de ces caractères ne peuvent être obtenus que dans l'un des modes d'impression internationaux de la DMP1.

Ainsi :

```
PRINT CHR$(&A0)
^
```

...fait apparaître à l'écran ^, tandis qu'avec :

```
PRINT #8,CHR$(&A0)
^
```

...on obtient ^ sur l'imprimante (alors que sur la DMP , l'accent circonflexe est codé **&5E**). En d'autres termes, le programme d'impression identifie **&A0** comme un code faisant partie de sa table de conversion et lui substitue **&5E**, afin que le caractère imprimé corresponde bien à celui qu'affiche l'écran. Quel que soit le mode linguistique choisi, le code **&5E** correspond toujours à l'accent circonflexe de la DPM1, ce qui n'est pas le cas pour tous les caractères contenus dans la table de conversion. Voici les autres caractères que contient la table de conversion :

CHR$	Caractère à l'écran	Transcodage imprimante	GB	U.S.A.	France	Allemagne	Espagne
&A0	^	&5E	^	^	^	^	^
&A2	••	&7B	†	†	†	†	••
&A3	£	&23	£	#	#	#	Pt
&A6	§	&40	†	†	†	§	†
&AE	¿	&5D	†	†	†	†	¿
&AF	¡	&5B	†	†	†	†	¡

Pour le caractère imprimé, se reporter à la page 18 du manuel d'instruction de la DMP .

La liste ci-dessus est extraite de la table des conversions par défaut. Pour les redéfinir, consulter le manuel de microprogrammation (SOFT 946).

Partie 11 : Les manettes de jeu

Le système d'exploitation du 6128 prévoit l'utilisation d'une ou de deux manettes de jeu. Au même titre que les touches du clavier, ces manettes peuvent être interrogées au moyen des instructions **INKEY** et **INKEY$**. Le bouton de commande principal de votre manette correspond le plus souvent à « Fire 2 ». Il est possible d'interroger directement la première et la seconde manette à l'aide des fonctions **JOY(Ø)** et **JOY(1)**. La fonction **JOY** renvoie un nombre dont la signification n'apparaît qu'en binaire, indiquant l'état des interrupteurs des manettes de jeu lors de la dernière interrogation du clavier.

Le tableau ci-dessous donne les valeurs transmises par les deux manettes. Dans la partie gauche figurent les valeurs obtenues par la fonction **JOY** et, dans la partie droite, celles à utiliser dans des expressions prenant les codes du clavier comme paramètres (fonctions **INKEY** et **KEY DEF**).

MANŒUVRE	Fonction JOY		VALEURS « CLAVIER »		
	BIT	VALEUR RENVOYEE	PREMIERE MANETTE	SECONDE MANETTE	TOUCHE CORRESP.
Haut	0	1	72	48	6
Bas	1	2	73	49	5
Gauche	2	4	74	50	R
Droite	3	8	75	51	T
Tir 2	4	16	76	52	G
Tir 1	5	32	77	53	F

Vous constatez que l'ordinateur assimile les valeurs renvoyées par la seconde manette à celles provenant des touches correspondantes du clavier (dernière colonne du tableau). Il est donc possible d'utiliser le clavier comme deuxième manette de jeu.

Partie 12 : Les formats de disquettes

Le BIOS prend en charge trois formats de disquettes différents : les formats SYSTEM, DATA ONLY et IBM. Lorsqu'un système fonctionnant sous AMSDOS accède à une disquette ne comportant aucun fichier ouvert, il en détecte automatiquement le format. Chaque format est doté à cette fin d'une numérotation de secteurs spécifique.

Bien que les disquettes « 3 pouces » soient à double face, seule la face supérieure est accessible par l'unité. Les deux faces peuvent être de format différent.

Caractéristiques communes à tous les formats

Simple face (les deux faces d'une disquette « 3 pouces » étant enregistrées séparément) :
Taille d'un secteur physique : 512 octets.
40 pistes numérotées de 0 à 39.
Taille du bloc CP/M : 1024 octets.
Répertoire : 64 entrées.

Le format SYSTEM

9 secteurs par piste, numérotés de &41 à &49.
2 pistes réservées.

Le chargement du CP/M n'étant possible qu'à partir d'une disquette de format système, ce dernier est donc le plus commun. Pour un amorçage à chaud, le CP/M 2.2 requiert également l'insertion d'une disquette de format système. Voici l'affectation des pistes réservées :

	CP/M 2.2	CP/M Plus
Piste 0 secteur &41	:amorçage	:amorçage
Piste 0 secteur &42	:configuration	: } inutilisé
Piste 0 secteurs &43 à &47	:inutilisés	: } inutilisé
Piste 0 secteurs &48 à &49	} CCP et BDOS	: } inutilisé
Piste 1 secteurs &41 à &49	} CCP et BDOS	: } inutilisé

Il existe une version du format système dans laquelle les pistes réservées ne contiennent aucun logiciel : le format VENDOR, destiné à la distribution des logiciels d'application.

Le format DATA ONLY

9 secteurs par piste, numérotés de &C1 à &C9.
Aucune piste réservée.

L'utilisation de ce format sous CP/M 2.2 est déconseillée dans la mesure où il n'autorise pas l'amorçage à chaud. Limité à AMSDOS et au CP/M Plus, il permet un léger gain de place sur la disquette.

Le format IBM (réservé au CP/M 2.2)

8 secteurs par piste, numérotés de &1 à &8.
1 piste réservée.

En conformité logicielle avec le format « simple face » utilisé par CP/M sur l'IBM PC. Le 6128 effectue la lecture et l'écriture des disquettes au format IBM, mais ne permet pas de les créer ou de les copier.

Partie 13 : Extensions résidentes du système (RSX)

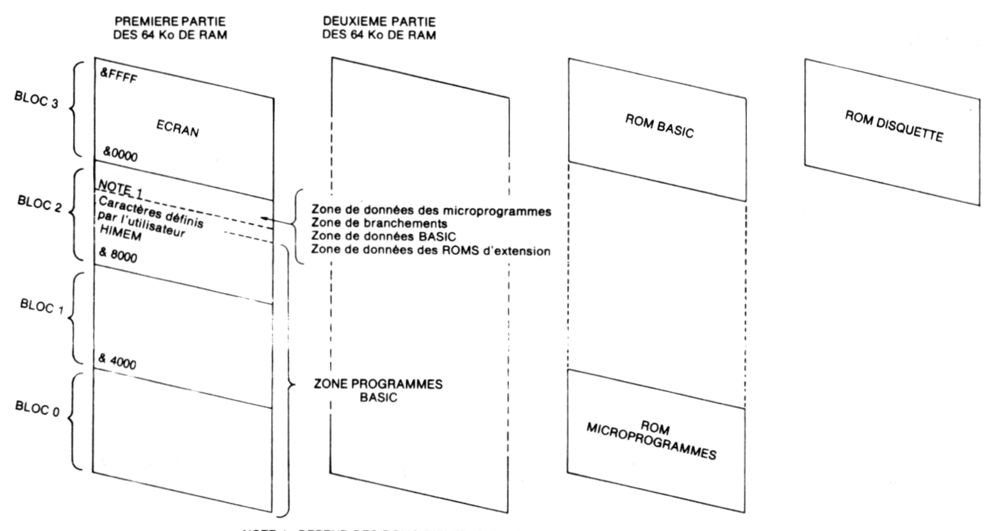
Nous avons évoqué les commandes externes au chapitre 5 (à propos de l'AMSDOS). Une commande externe sert à élargir le répertoire du BASIC en lui ajoutant de nouvelles commandes indiquées par le préfixe |. Les instructions machine des nouvelles commandes AMSDOS résident en mémoire ROM (mémoire morte) et la procédure d'adjonction des commandes est assurée automatiquement dès que le 6128 démarre en BASIC.

Il est également possible d'ajouter d'autres commandes (après lancement du BASIC) en chargeant les instructions machine en mémoire RAM (mémoire vive). Ces nouvelles commandes sont appelées RSX (extensions résidentes) et fonctionnent exactement comme les extensions stockées dans la ROM. Les extensions résidentes doivent être chargées à partir de la disquette (ou de la cassette) à chaque initialisation ou réinitialisation du 6128 en BASIC. Elles servent habituellement à commander des périphériques intelligents tels que des crayons optiques ou des synthétiseurs de parole.

Le chapitre 8 décrit l'utilisation des extensions résidentes RSX pour accéder au second bloc des 64 Ko de mémoire.

Partie 14 : La mémoire

Le 6128 possède une mémoire RAM de 128 Ko et une ROM de 48 Ko, accessibles au BASIC 1.1 (voir figure ci-dessous). Les premiers 64 Ko de RAM sont divisés en quatre blocs de 16 Ko chacun, numérotés de 0 à 3. L'écran exploite le bloc 3. La partie supérieure du bloc 2 contient les variables système.

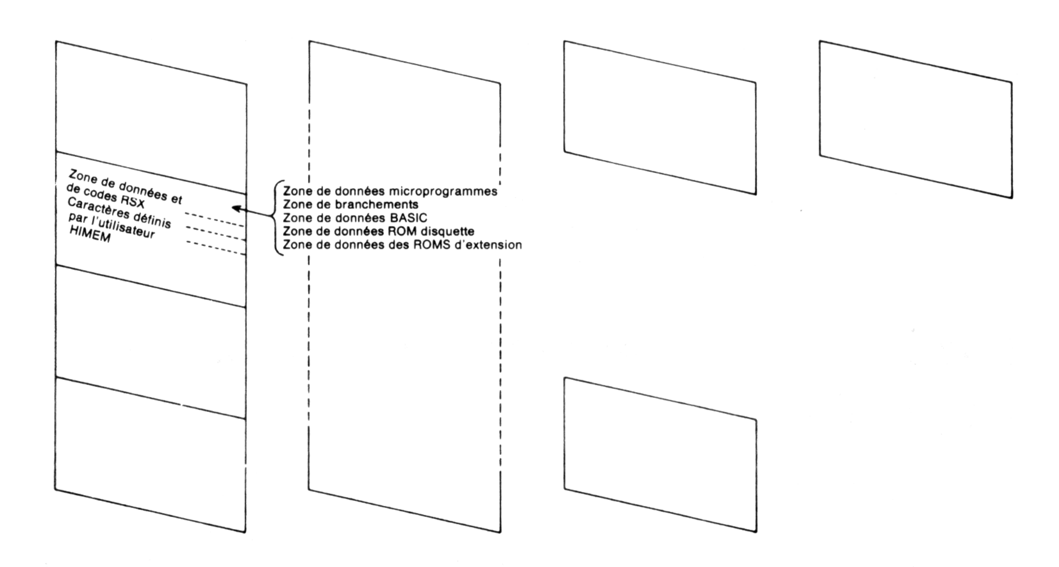


Topographie de la mémoire pour le BASIC 1.1

Au départ, les caractères définis par l'utilisateur se trouvent immédiatement au-dessus de la limite HIMEM, mais celle-ci peut être modifiée par une commande **MEMORY** et elle est automatiquement abaissée de 4 Ko lors de l'ouverture de fichiers AMSDOS pour création d'une mémoire tampon. Le nombre de caractères définis par l'utilisateur ne peut être modifié que si aucun changement du jeu de caractères n'est intervenu depuis sa détermination précédente (à moins qu'une commande **SYMBOL AFTER 256** n'ait fixé « pas de caractères définis par l'utilisateur »). Lors du lancement du BASIC, les caractères définis par l'utilisateur sont fixés comme si une commande **SYMBOL AFTER 240** avait été envoyée.

Il est donc prudent de libérer la zone des caractères définis par l'utilisateur avant de modifier HIMEM définitivement, puis de restaurer cette zone dans sa nouvelle position. Les programmes suivants pourront ainsi modifier l'affectation faite par la commande **SYMBOL AFTER**.

L'exemple ci-dessous illustre cette façon de procéder lorsque HIMEM est abaissée par le chargement d'extensions résidentes RSX.



Topographie de la mémoire avec RSX à l'emplacement recommandé

Entrées/Sorties supplémentaires

Les adresses des ports d'Entrée/Sortie sont, pour la plupart, réservées par l'ordinateur. Les adresses inférieures à &7FFF, notamment, sont absolument prohibées.

Les lignes d'adresses A0 à A7 seront affectées à la désignation du type de périphérique et les lignes A8 et A9 à la sélection de l'un des registres du périphérique sollicité. Parmi les lignes restantes seul A10 doit être décodée (état bas) alors que les lignes A11 jusqu'à A15 doivent être à l'état haut. Aux registres de chaque périphérique seront ainsi affectées des adresses du type &F8??, &F9??, &FA?? et &FB??, où les deux derniers symboles désigneront soit une interface de communication (de DC à DF) soit tout autre type de périphérique (de EO à FE).

On utilisera les instructions Z80 recopiant le registre B dans la moitié supérieure du bus d'adresses (A15 à A8).

ROMs auxiliaires

Il est possible d'adresser des ROMs auxiliaires en plus de la ROM interne. La logique d'arbitrage d'adresses et de sélection par bloc sera alors située dans un module connecté au bus d'extension, l'ordinateur fournissant tous les signaux nécessaires à ce dernier.

Partie 15 : L'émulateur de terminaux du CP/M Plus

Dans la première partie de ce chapitre figure un tableau de caractères de contrôle (avec leur action). Ces actions s'effectuent lorsque le BASIC ou le CP/M 2.2 envoie le texte à l'écran. Elles ont été choisies pour leur simplicité et pour leur fidélité aux fonctions du logiciel de génération de caractères. Elles sont propres aux ordinateurs Schneider et leur logiciel ne peut donc pas être adapté.

Afin de permettre la portabilité du logiciel CP/M Plus d'un ordinateur à l'autre pour un environnement commercial, il est normal de doter celui-ci d'une diversité de fonctions de traitement de texte standard. L'implémentation du CP/M Plus sur 6128 intègre un émulateur de terminaux disposant de fonctions très similaires à celles d'un terminal Zénith Z19/Z29. La procédure d'installation du logiciel CP/M Plus comprend généralement une option standard pour ce type de terminaux.

Les fonctions de l'émulateur de terminaux du CP/M Plus couvrent une grande partie de celles du logiciel de génération de caractères ; il faut noter toutefois que les codes de contrôle sont alors différents.

Les caractères compris entre &20 et &FF sont affichés à la position courante du curseur. Si le curseur n'est pas dans la dernière colonne de l'écran, il se déplace d'une colonne vers la droite. S'il se trouve dans cette colonne et que la fonction de retour automatique à la ligne est activée, il se place à gauche de la ligne suivante, en faisant remonter le texte affiché si nécessaire.

Les caractères de contrôle (compris entre &00 et &1F) sont interprétés comme suit :

&07	BEL	Signal sonore.
&08	BS	Espace arrière. Déplace le curseur d'une colonne vers la gauche. Si le curseur se trouve sur la première colonne, mais pas sur la première ligne, et que la fonction de retour automatique à la ligne est activée, il se place sur la dernière colonne de la ligne supérieure.
&OA	LF	Saut de ligne. Fait descendre le curseur d'une ligne, et remonter le texte si nécessaire.
&OD	CR	Retour chariot. Place le curseur sur la première colonne de la même ligne.
&1B	ESC	Introduit une séquence de changement de mode.

Tous les autres codes de contrôle sont ignorés.

Les séquences de changement de mode ci-dessous sont reconnues. Tout autre caractère suivant le caractère ESC est affiché en déplaçant le curseur, ce qui permet d'afficher les caractères correspondant aux codes de contrôle &00 à &1F. Dans de nombreux langages d'application le code de contrôle &09(TAB) correspond à des espaces et la séquence [ESC][TAB] n'affiche donc pas le caractère qui lui correspond.

[ESC]0 — Désactive la ligne d'état. Les messages système sont visualisés par une sortie écran ordinaire, la ligne d'état devient disponible.

[ESC]1 — Active la ligne d'état. Les messages système apparaissent alors sur la ligne inférieure de l'écran.

[ESC]2 <n> — Change le jeu de caractères (voir partie 16 de ce chapitre). n est le paramètre de la langue choisie, masqué par le code &07. Certaines matrices de caractères comprises entre &20 et &7F sont remplacées par d'autres caractères compris entre &80 et &FF. Cette commande agit de façon très semblable à la commande de contrôle des imprimantes possédant des jeux de caractères définissables par logiciel.

<n> = 0	Etats-Unis
<n> = 1	France
<n> = 2	Allemagne
<n> = 3	Royaume-Uni
<n> = 4	Danemark
<n> = 5	Suède
<n> = 6	Italie
<n> = 7	Espagne

[ESC]3 <m>	Change le mode de l'écran. <m> = mode écran + &20. Cette valeur est masquée par le code &3 pour donner les modes 0 à 2. Le mode 3 est ignoré. L'écran s'efface mais le curseur ne change pas de place.
[ESC]A	Déplace le curseur vers le haut sauf s'il se trouve sur la première ligne.
[ESC]B	Déplace le curseur vers le bas sauf s'il se trouve sur la dernière ligne.
[ESC]C	Déplace le curseur vers la droite sauf s'il se trouve sur la dernière colonne.
[ESC]D	Déplace le curseur vers la gauche sauf s'il se trouve sur la première colonne.
[ESC]E	Efface la page. Le curseur ne change pas de place. Cette commande efface la totalité de l'écran, même en mode 24 x 80. (Les autres codes ESC n'affectent que la zone 24 x 80 lorsque le mode 24 x 80 est sélectionné.)
[ESC]H	Déplace le curseur jusqu'à la première colonne de la première ligne.
[ESC]I	Déplace le curseur d'une ligne vers le haut, en faisant descendre le texte si nécessaire.
[ESC]J	Efface le texte jusqu'à la fin de la page, caractère sous le curseur compris. Le curseur ne change pas de place.
[ESC]K	Efface le texte jusqu'à la fin de la ligne, y compris le caractère situé sous le curseur. Le curseur ne change pas de place.
[ESC]L	Insère une ligne. Toutes les lignes situées sous la ligne du curseur, celle-ci comprise, descendent. La ligne se situant à la position du curseur est alors vide. La position du curseur ne change pas.
[ESC]M	Supprime une ligne. Toutes les lignes situées sous la ligne du curseur, celle-ci comprise, remontent. La ligne du bas de l'écran est alors vide. La position du curseur ne change pas.
[ESC]N	Supprime un caractère. Tous les caractères situés à droite du curseur se déplacent d'une colonne vers la gauche, laissant un espace en fin de ligne. La position du curseur ne change pas.
[ESC]Y <l c>	Déplace le curseur jusqu'à la position indiquée. Si celle-ci se trouve à l'extérieur de l'écran, le curseur se place au bord de l'écran. <l> = ligne + &20 et <c> = colonne + &20. L'angle supérieur gauche de l'écran correspond à la ligne 0, colonne 0.

[ESC]b <pc> Définit la couleur du texte et affecte tous les caractères de l'écran. <pc> est le paramètre de la couleur, masqué par le code &3F et traité ensuite comme trois nombres de 2 bits, indiquant chacun l'intensité d'une des trois couleurs primaires : bits 0,1 pour le bleu, bits 1,2 pour le rouge et bits 3,4 pour le vert. Le 6128 permet trois intensités représentées en binaire comme suit :

CPC 6128	Intensité zéro	Demi-intensité	Pleine intensité
Bits de couleur	00	01 ou 10	11

[ESC]c <pc> Définit la couleur de fond de l'écran. Affecte tout le fond et la bordure de l'écran. Les couleurs sont spécifiées comme l'indique le tableau ci-dessus.

[ESC]d Efface le texte jusqu'au début de la page, y compris le caractère situé sous le curseur. La position du curseur ne change pas.

[ESC]e Désactive la trace du curseur pour un meilleur confort visuel. Le curseur réapparaît 1/10 seconde après l'affichage du dernier caractère entré.

[ESC]f Réactive la trace du curseur.

[ESC]j Sauvegarde la position du curseur.

[ESC]k Restitue la position du curseur sauvegardée par [ESC]j.

[ESC]l Efface la ligne. La position du curseur ne change pas.

[ESC]o Efface les caractères jusqu'au début de la ligne, y compris celui qui se trouve sous le curseur. La position du curseur ne change pas.

[ESC]p Etablit le mode vidéo inverse. Les couleurs de fond et de texte des caractères imprimables sont inversées.

[ESC]q Sort du mode vidéo inverse.

[ESC]r Active le mode soulignement (non disponible sur le 6128).

[ESC]u Désactive le mode soulignement (non disponible sur le 6128).

[ESC]v Le curseur passe en fin de ligne.

[ESC]w — Supprime les caractères jusqu'à la fin de la ligne.

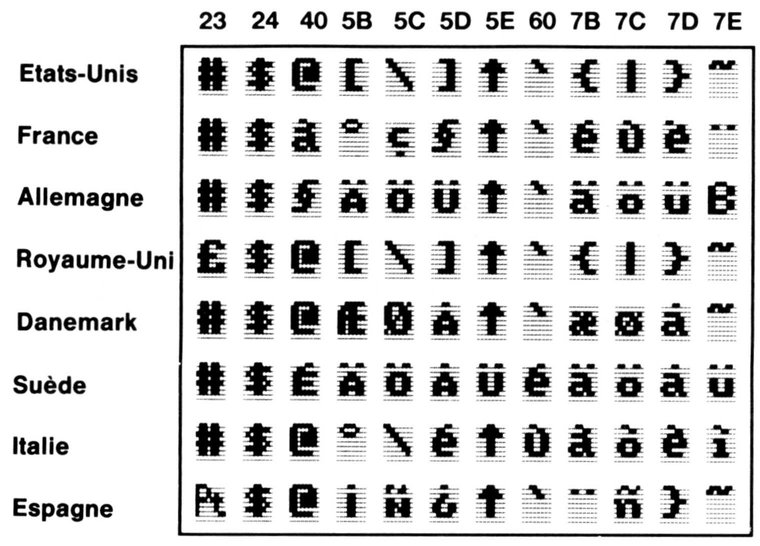
[ESC]x — Etablit le mode 24 x 80. Pour certains programmes d'application, un écran standard 24 x 80 est nécessaire. Cette commande active ce mode sans tenir compte de la taille de l'écran, qui dépend de l'ordinateur, du pays d'utilisation et de l'activation de la ligne d'état. Le contenu de l'écran s'efface.

[ESC]y — Désactive le mode 24 x 80. Le contenu de l'écran s'efface.

Partie 16 : Le jeu de caractères du CP/M Plus

Dans la partie 10 de ce chapitre figure une table de conversion des caractères du BASIC et du CP/M 2.2 sous une forme correspondant à la langue sélectionnée par l'imprimante. Cette possibilité de transcodage est cependant limitée, car le jeu de caractères du BASIC comprend très peu de caractères étrangers par rapport aux imprimantes.

Cette conversion opère également sous CP/M Plus mais le jeu de caractères a été amélioré pour permettre une correspondance quasi totale entre caractères affichés et caractères imprimés (le seul symbole manquant à l'écran est le symbole monétaire suédois remplaçant le $). La table ci-dessous confirme l'étendue de la correspondance.



	23	24	40	5B	5C	5D	5E	60	7B	7C	7D	7E
Etats-Unis	#	$	@	[	\	]	↑	`	{	|	}	~
France	#	$	à	°	ç	§	↑	`	é	ù	è	¨
Allemagne	#	$	§	Ä	Ö	Ü	↑	`	ä	ö	ü	ß
Royaume-Uni	£	$	@	[	\	]	↑	`	{	|	}	~
Danemark	#	$	@	Æ	Ø	Å	↑	`	æ	ø	å	~
Suède	#	$	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Italie	#	$	@	°	\	é	↑	ù	à	ò	è	ì
Espagne	Pt	$	@	¡	Ñ	¿	↑	`	¨	ñ	}	~

Jeu de caractères international du CP/M Plus

Voici la marche à suivre pour exploiter l'ordinateur dans une langue autre que l'américain :

1. Sélectionner sur l'imprimante la langue désirée (le plus souvent à l'aide de micro-interrupteurs DIP ; certaines imprimantes acceptent cependant les codes de contrôle).

2. Sélectionner le jeu de caractères écran correspondant, par la commande :

   LANGUAGE <n>

ou par envoi du code

   [ESC]2<n>

à l'émulateur de terminaux.

En pratique, cette initialisation peut s'effectuer dans le cadre de la procédure **PROFILE.-SUB**, à l'aide des fonctions **LANGUAGE** et **SETLST**. A l'expédition, le CP/M Plus est configuré en version américaine, essentiellement parce que le clavier dispose du signe # sur la touche 3. Le clavier américain est très souvent utilisé pour le traitement de texte.

Logiciel 7 bits...

Bien que très puissante, cette fonction, qui remplace les caractères « normaux » (américains) par ceux de la langue choisie, ne permet plus d'afficher ces caractères. Il s'agit là d'un aléas courant et inévitable des logiciels à 7 bits. Presque tous les logiciels (même les utilitaires du CP/M Plus, les traitements de texte) opérent sur un jeu de caractères à 7 bits seulement. Pour le Royaume-Uni, la disparition du signe # au profit du symbole monétaire £ est plutôt bien acceptée, surtout en traitement de texte ; quant au listage des programmes, **LIST £ 8**, par exemple, l'utilisateur s'habitue à faire la conversion mentalement.

Malheureusement, pour les autres langues, certains caractères remplacent la barre verticale et les crochets. C'est bien sûr plus pratique pour la lisibilité des caractères accentués, mais pour les programmes d'application nécessitant ces signes, **DIR [ALL]**, par exemple, la commodité de lecture et de frappe (si les touches sont modifiées) s'en trouve nettement réduite. N'oubliez pas que les programmes travaillent en valeurs ASCII, qui se soucient peu des caractères affichés à l'écran et, avec un logiciel à 7 bits, celles-ci ne sont pas assez nombreuses.

Exploitation du système avec des jeux de caractères à 8 bits

Le jeu de caractères BASIC possède 256 symboles différents ; les valeurs 128 à 255 (&80 à &FF) correspondent aux différents symboles graphiques pour les jeux et les applications individuelles (personnages, c̀ur/carreau/pique/trèfle, etc.). Le CP/M Plus possède également 256 caractères, mais les 128 derniers diffèrent de ceux du BASIC : ils apportent à l'environnement du CP/M Plus une touche à caractère international et professionnel.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	∞	⊝	Γ	Δ	⊗	×	÷	∴	∏	↓	Σ	←	→	±	↔	Ω
1	α	β	γ	δ	ε	θ	λ	μ	π	ρ	σ	τ	φ	χ	ψ	ω
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	◊

Caractères 0 à 127 (&00 à &7F)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	■	╧	╟	╚	╤	║	╔	╠	╢	╝	═	╩	╗	╣	╦	╬
9	·	╵	╴	└	╷	│	┌	├	╶	┘	─	┴	┐	┤	┬	┼
A	ª	º	°	£	©	¶	§	†	¼	½	¾	«	»	₧	¿	¡
B	ƒ	¢	¨	´	^	‰	⅛	⅜	⅝	⅞	ß	○	•	¥	®	™
C	Á	É	Í	Ó	Ú	Â	Ê	Î	Ô	Û	À	È	Ì	Ò	Ù	Ÿ
D	Ä	Ë	Ï	Ö	Ü	Ç	Æ	Å	Ø	Ñ	Ã	Õ	≥	≤	≠	≃
E	á	é	í	ó	ú	â	ê	î	ô	û	à	è	ì	ò	ù	ÿ
F	ä	ë	ï	ö	ü	ç	æ	å	ø	ñ	ã	õ	⇒	⇐	⇔	≡

Caractères 128 à 255 (&80 à &FF)

Jeu de caractères standard du CP/M Plus

Les logiciels fonctionnant avec des caractères de 8 bits peuvent exploiter ce jeu de caractères pour tracer des cases, comme le montre la figure ci-dessus, mais aussi pour accéder directement a toutes les langues sans modification spécifique de clavier. Les caractères tels que les barres verticales et les crochets sont également disponibles.

Les logiciels à 8 bits sont encore très rares et les caractères 0 à 31 et 128 à 255 n'apparaîtront que rarement à l'écran tels qu'ils figurent ci-dessus. Les imprimantes, par contre, restitueront ces symboles plus fidèlement.

Chapitre 8
Complément d'information sur le Gestionnaire de Blocs BANK MANAGER

Les extensions du BASIC permettent d'accéder au second bloc des 64 KO de RAM.

Sujets abordés:

* Stockage des images écran
* Fonctionnement des fichiers virtuels

Comme l'indique la topographie de la mémoire du BASIC 1.1 (représentée à la partie 14 du chapitre 7), sur les 128Ko de RAM, 64Ko restent inutilisés. Le BASIC et les programmes système résident dans la ROM qui, ajoutée à la ROM disquette, étend la mémoire de 64Ko à 112Ko (64Ko de RAM, 48Ko de ROM).

Chaque lot de 16Ko s'appelle un « bloc » et quatre blocs (soit 64ko) forment un banc. C'est pourquoi la technique de sélection des blocs se nomme « changement de bancs ».

Le microprocesseur Z80 ne peut gérer que 64Ko à la fois, aussi le système d'exploitation contient-il des instructions qui sollicitent la ROM des microprogrammes, plutôt que le bloc 0 de la RAM, et la ROM du BASIC ou de la disquette, plutôt que le bloc 3. Ce changement de blocs intervient automatiquement lorsque le BASIC ou les microprogrammes sont requis. Ce procédé est également appliqué à la RAM pour favoriser le recouvrement de la RAM plutôt que celui de la ROM. Le changement est assurée par un programme rédigé en assembleur.

Le programme **BANKMAN.BAS** réside sur la face 1 d'une des disquettes système. S'il est exécuté après lancement du BASIC, il instaure le code RSX de gestion de blocs, d'où son appellation de « gestionnaire de blocs ».

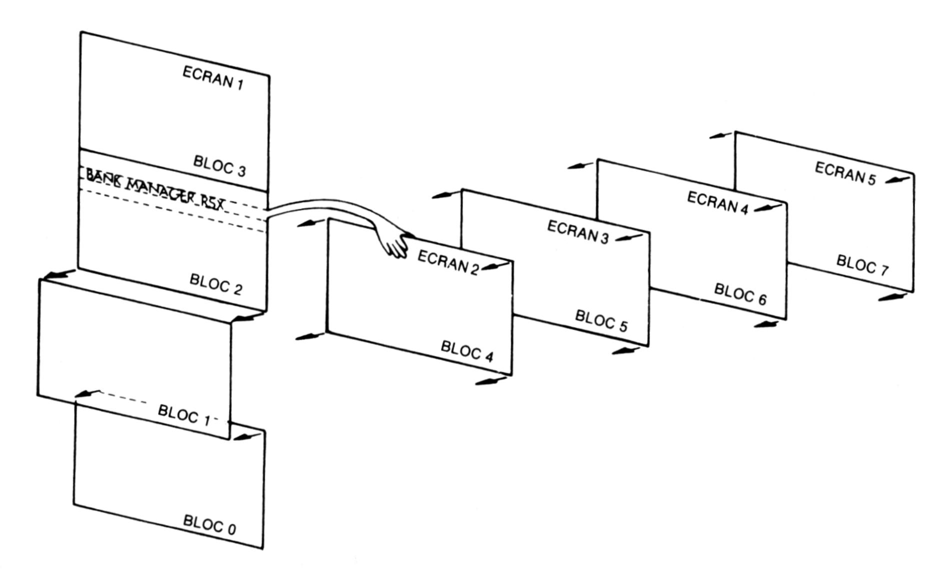
La deuxième partie des 64Ko de mémoire sert notamment d'espace provisoire de stockage des écrans graphiques. Elle permettra par exemple de stocker un programme de dessin tel que « Salut l'artiste » possédant plusieurs écrans différents, ou un jeu vidéo comportant plusieurs écrans déjà prêts.

Cette partie de la mémoire permet aussi d'étendre l'espace de travail des variables, qui peut être considéré comme une extension de l'espace réservé aux chaînes, ou tout simplement comme disque virtuel (sur RAM).

Partie 1 : Stockage des images d'écran

Choisissez votre écran...

BANK MANAGER peut mettre le bloc 1 de côté et sélectionner à sa place l'un des quatre blocs des derniers 64Ko de la mémoire, comme l'illustre le diagramme ci-dessous. Vous remarquerez que chaque bloc de cette partie de la mémoire occupe la même adresse (&4000 à &7FFF). Le contenu du bloc 1 (probablement le bloc central de votre programme en BASIC) est préservé et restitué lorsque BANK MANAGER a terminé. Il existe trois autres sélections de bancs possibles (outre les cinq représentées ci-dessous) mais elles ne servent qu'à implémenter le CP/M Plus.



Changement matériel de bancs

BANK MANAGER gère deux commandes de déplacement d'écrans d'un bloc à l'autre. Les blocs 4 à 7 sont sélectionnés selon les besoins.

La commande:

|**SCREENSWAP**,[<**zone écran**>,] <numéro d'écran>, <numéro d'écran>

intervertit le contenu des deux blocs.

La commande:

|**SCREENCOPY**,[<**zone écran**>,] <numéro d'écran cible>, <numéro d'écran source>

copie le contenu d'un bloc dans un autre bloc.

Le paramètre facultatif <zone écran> provoque la copie d'1/64 du bloc (soit 256 octets des 16Ko). Il peut varier de 0 à 63. Ce mode permet d'effectuer une autre tâche entre deux opérations de copie. L'échange de contenus d'écrans dure environ 150/300 secondes (soit 150 unités TIME).

Le premier <numéro d'écran> doit être 1 (écran normal) et le second peut prendre les valeurs 2, 3, 4 ou 5. Les copies ou échanges d'écrans faisant intervenir l'écran 1 comme cible ou comme source sont bien plus rapides. Comme pour la copie d'écrans sur disquette, il convient de veiller au défilement d'écran généré par le matériel. Il faut pour cela construire et visualiser toutes les images écran avec la même position matérielle que celle de l'écran 1, le plus simple étant de faire appel à la commande **MODE.**

Pratique des commandes de changement d'écrans

Commencez par lancer le programme BANK MANAGER à partir de la face 1 d'une des disquettes système:

```
RUN "BANKMAN"
```

Tapez ensuite:

```
MODE 1
```

Le contenu de l'écran s'efface. Tapez alors:

```
'Ceci est l'ecran original
|SCREENCOPY,3,1 'Stocke l'ecran original en memoire No 3
CLS
```

Le contenu de l'écran s'efface de nouveau. Tapez:

```
'Ceci est l'ecran intermediaire
|SCREENCOPY,2,1 ' Envoi l'ecran intermediaire en mem 2
|SCREENSWAP,2,3 ' Echange la memoire 2 et la memoire 3
|SCREENCOPY,1,3 ' Fait venir a l'ecran la memoire 3
|SCREENCOPY,1,2 ' Fait venir a l'ecran la memoire 2
```

Pour terminer, la dernière partie du chapitre 9 comprend un programme complet, « Salut l'artiste », qui met en jeu les fonctions de changement d'écrans de BANK MANAGER.

Partie 2: Fonctionnement des fichiers virtuels

Encore ces maudits fichiers ...

Pris comme disque virtuel, les derniers 64Ko de mémoire sont divisés en plusieurs enregistrements de longueur fixe, constituant un fichier virtuel (de mémoire RAM). Cette longueur doit être comprise entre 0 et 255 octets, 2 octets étant le minimum recommandé. Après avoir fixé la longueur de chaque enregistrement virtuel, vous pourrez y accéder grâce à son numéro. Rien ne vous empêche d'écrire le fichier virtuel avec telle ou telle longueur d'enregistrement, et de le lire avec une longueur différente.

Remarque : Ce fichier virtuel ne peut contenir que des données. En effet, rien n'est prévu pour qu'il puisse contenir des instructions de programmes exécutables.

Comme pour les fichiers à accès sélectif, ce fichier virtuel reprend la notion de numéro d'enregistrement courant. Ceci permet l'utilisation d'un numéro d'enregistrement par défaut, particulièrement utile pour les recherches à l'intérieur du fichier.

La commande :

| **BANKOPEN**, <longueur d'enregistrement virtuel>

fixe la longueur de tous les enregistrements, ainsi que la valeur zéro pour l'enregistrement courant, mais N'EFFACE AUCUNEMENT le contenu de la mémoire.

La commande :

| **BANKWRITE**,@ <code retour>,<chaîne alphanumérique> [,<numéro d'enregistrement virtuel>]

écrit la <chaîne alphanumérique> dans le fichier virtuel.

Le <numéro d'enregistrement virtuel> définit l'enregistrement en cours d'écriture. S'il est omis, ce paramètre est remplacé par le numéro d'enregistrement courant. Le pointeur d'enregistrement passe ensuite à l'enregistrement suivant.

Si la <chaîne alphanumérique> ne remplit pas entièrement l'enregistrement, les anciens caractères (non écrasés) restent en fin d'enregistrement. Si la chaîne est plus longue que l'enregistrement, les caractères en excédent sont supprimés pour éviter leur « débordement » dans l'enregistrement suivant.

Le <code retour> est une variable entière correspondant au numéro d'enregistrement qui vient d'être écrit (si l'opération a abouti). Si l'opération a échoué, pour une raison quelconque, ce code est négatif.

-1 fin de fichier, indique que l'adresse du numéro d'enregistrement demandé dépasse 64Ko.

-2 erreur lors d'un changement de bloc, ce qui ne devrait jamais se produire.

Exemples:

```
|BANKOPEN,10
|BANKWRITE,@r%,"123 test",0
|BANKWRITE,@r%,w$
```

La commande:

| **BANKREAD**,@ <code retour>,@ <chaîne alphanumérique> [,<numéro d'enregistrement virtuel>]

lit un enregistrement et le copie dans la chaîne à partir du fichier virtuel.

Le <numéro d'enregistrement virtuel> définit l'enregistrement à lire. S'il est omis, ce paramètre est remplacé par le numéro d'enregistrement courant. Le pointeur d'enregistrement passe ensuite à l'enregistrement suivant.

Si le contenu de l'enregistrement ne remplit pas entièrement la chaîne, les anciens caractères (non écrasés) restent à la fin de la chaîne. Si ce contenu dépasse la longueur de la chaîne, les caractères en excédent sont supprimés car la longueur d'une chaîne ne peut pas être augmentée au cours d'une commande externe.

Le <code retour> est une variable entière correspondant au numéro d'enregistrement qui vient d'être lu (si l'opération a abouti). Si l'opération a échoué, pour une raison quelconque, ce code est négatif.

-1 fin de fichier, indique que l'adresse du numéro d'enregistrement demandé dépasse 64Ko.

-2 erreur lors d'un changement de bloc, ce qui ne devrait jamais se produire.

Exemple:

```
|BANKREAD,@r%,i$,0
```

Recherche

Il est possible de rechercher une chaîne particulière à l'intérieur des enregistrements stockés.

La commande:

> | **BANKFIND**,@ <code retour>,<chaîne recherchée> [,<numéro d'enregistrement de départ> [,<numéro d'enregistrement de fin>]]

passe tous les enregistrements virtuels en revue. Le <numéro d'enregistrement de départ> spécifie le début de la recherche. S'il est omis, la recherche commence à l'enregistrement courant.

La recherche procède enregistrement par enregistrement, à l'intérieur des 64 derniers Ko de la mémoire, jusqu'à trouver la chaîne désirée.

Si un <numéro d'enregistrement de fin> a été spécifié, la recherche s'arrête après examen de cet enregistrement, à moins que la chaîne n'ait été trouvée avant d'atteindre cet enregistrement.

Si la recherche aboutit, le pointeur prend la valeur du numéro d'enregistrement contenant la chaîne recherchée (sinon celui-ci ne change pas).

Le <code retour> est une variable entière correspondant au numéro d'enregistrement contenant la chaîne recherchée (si l'opération a abouti). Si l'opération a échoué, pour une raison quelconque, le code de retour est négatif.

-1 fin de fichier, indique que l'adresse du numéro d'enregistrement de départ dépasse 64Ko ou qu'elle est supérieure à celle de l'enregistrement de fin de recherche.

-2 erreur lors d'un changement de bloc, ce qui ne devrait jamais se produire.

-3 pas de chaîne retrouvée.

La <chaîne recherchée> peut contenir des jokers, indiqués par des caractères nuls: **CHR$(0)**, et la comparaison s'effectue par rapport à la <longueur d'enregistrement virtuel> ou à la longueur de la <chaîne recherchée>, si celle-ci est plus courte.

Exemples:

```
|BANKFIND,@r%,"123 test",0
|BANKFIND,@r%,f$,100,200
```

Ne ratez pas la correspondance

Les erreurs évidentes, telles qu'un nombre de paramètres erroné, sont signalées par le message d'erreur « **Bad command** » (commande erronée). Cependant, les commandes externes ne détectent pas les erreurs de « **Type mismatch** » (type de variable ne correspondant pas). L'utilisateur doit donc veiller à utiliser des paramètres corrects.

Le programme ci-dessous exploite les commandes de disque virtuel pour établir et interroger une base de données comportant des anagrammes de mots de 7 lettres. Il vous donne les mots qui conviennent et autorise, en outre, l'emploi de jokers.

Par exemple, les anagrammes de **FIGURES** qui se présentent sous la forme: **?RUGS??** (les deux derniers points d'interrogation peuvent éventuellement être supprimés) sont **FRUGSIE**, **FRUGSEI**, **IRUGSFE**, **IRUGSEF**, **IRUGSFI** et **IRUGSIF**.

Il faut un certain temps au programme pour établir la base de données, mais la taille de la mémoire à gérer, 64Ko, le justifie.

```
10 ' ANAGRAMMES par ROLAND PERRY
20 ' copyright (c) AMSOFT 1985
30 '
40 ' Rappelez vous de faire RUN"BANKMAN"     avant de lancer
 votre programme
50 '*********************************************************
60 '
70 MODE 2
80 DEFINT a-z
90 r%=0:|BANKOPEN,7
100 INPUT "Donnez un mot de 7 lettres";s$
110 IF LEN(s$)<>7 THEN 100
120 PRINT " patientez..."
130 LOCATE 1,5:PRINT "computing:"
140 FOR c1=1 TO 7
150 FOR c2=1 TO 7
160 IF c2=c1 THEN 370
170 FOR c3=1 TO 7
180 IF c3=c2 OR c3=c1 THEN 360
190 FOR c4=1 TO 7
200 IF c4=c3 OR c4=c2 OR c4=c1 THEN 350
210 FOR c5=1 TO 7
220 IF c5=c4 OR c5=c3 OR c5=c2 OR c5=c1 THEN 340
230 FOR c6=1 TO 7
240 IF c6=c5 OR c6=c4 OR c6=c3 OR c6=c2 OR c6=c1 THEN 330
250 FOR c7=1 TO 7
260 IF c7=c6 OR c7=c5 OR c7=c4 OR c7=c3 OR c7=c2 OR c7=c1 TH
EN 320
```

```
270 o$=MID$(s$,c1,1)+MID$(s$,c2,1)+MID$(s$,c3,1)+MID$(s$,c4,
1)+MID$(s$.c5.1)+MID$(s$,c6,1)+MID$(s$,c7,1)
280 LOCATE 12,5:PRINT x;o$
290 |BANKWRITE,@r%,o$
300 IF r%<0 THEN STOP
310 x=x+1
320 NEXT c7
330 NEXT c6
340 NEXT c5
350 NEXT c4
360 NEXT c3
370 NEXT c2
380 NEXT c1
390 lastrec=r%
400 ' pour les visualiser
410 r%=0:g$=SPACE$(7)
420 PRINT:INPUT "ANAGRAMME a RETROUVER (Utilisez <?> comme j
ocker): ",m$
430 m$=LEFT$(m$,7)
440 FOR x=1 TO LEN(m$)
450 IF MID$(m$.x.1)="?" THEN MID$(m$.x.1)=CHR$(0)
460 NEXT
470 |BANKFIND,@r%,m$,0,lastrec
480 IF r%<0 THEN GOTO 420
490 |BANKREAD,@r%,g$
500 PRINT g$,
510 |BANKFIND,@r%,m$,r%+1,lastrec
520 GOTO 480
```

Chapitre 9
A vos heures de loisir...

Ce chapitre aborde en douceur certaines notions fondamentales de l'informatique en général et du 6128 en particulier. Bien qu'il ne soit pas indispensable de le lire avant d'utiliser votre ordinateur, il pourra vous aider à mieux comprendre ce qui se passe « sous le capot ».

Partie 1 : Généralités

Feu sur les envahisseurs de l'espace !

Même si votre seule passion des jeux vidéo motive votre achat du 6128, vous êtes peut-être curieux de connaître ce que recouvrent les termes. Le matériel est constitué de tout ce que vous pouvez toucher (le clavier, l'écran, les câbles de raccordement, etc.), le reste répondant à l'appellation de « logiciel » (programmes, manuels et informations sur disque ou cassette).

Certaines actions de l'ordinateur dépendent du matériel, mais c'est au logiciel de les utiliser pour produire des textes et des graphiques à l'écran.

Le matériel dirige le canon à électrons sur l'affichage électroluminescent du moniteur, le logiciel ordonne et distribue l'émission afin de produire des séquences significatives sur l'écran : l'effet de décollage du vaisseau spatial, ou l'apparition d'une lettre à l'écran lorsque vous appuyez sur la touche correspondante du clavier.

Question : Pourquoi un ordinateur est-il meilleur qu'un autre ?

Un matériel sans logiciel ne vaut rien, et inversément. Il ressort donc que la valeur des performances d'un ordinateur ne peut venir que de l'intégration harmonieuse des deux, permettant de réaliser différentes tâches. Il existe plusieurs moyens de juger les performances logicielles et matérielles d'une machine.

Les critères généralement admis pour qualifier les ordinateurs personnels sont actuellement :

1. La résolution de l'écran : le plus petit élément visible de l'affichage

Elle résulte d'une combinaison d'éléments différents : nombre de couleurs disponibles, nombre de pixels (petits points), nombre de caractères de texte définissables par écran. Vous constaterez que le 6128 dispose de caractéristiques extrêmement intéressantes dans ce domaine.

2. L'interpréteur BASIC

Presque tous les ordinateurs familiaux possèdent un interpréteur BASIC qui permet de se lancer rapidement dans la conception de programmes. Le BASIC intégré à la machine est lui-même un programme, mais extrêmement élaboré et compliqué, ayant nécessité des millions d'heures de travail pour sa mise au point, depuis son « invention » aux USA. Le « Beginners All Purpose Symbolic Instruction Code » (langage symbolique tous usages pour débutants) est le plus utilisé de l'informatique personnelle et, comme tous les langages, il possède de nombreux « dialectes ».

Le BASIC disponible sur le 6128 est l'un des plus compatibles et peut utiliser des progiciels écrits sous le système d'exploitation de disquette CP/M. C'est une implémentation très rapide du BASIC et bien qu'il ne vous semble pas qu'il y ait une grande différence entre 0,05 et 0,125 seconde, sur des milliers d'instructions à traiter, le même programme sera presque trois fois plus rapide d'une version à l'autre.

Vous entendrez souvent parler de langage machine : c'est le seul langage que la machine comprenne directement ; il est très rapide mais désespérément long à programmer, c'est pourquoi on utilise le BASIC couplé à un interpréteur pour la traduction en langage machine.

Le BASIC de votre Schneider est un des plus rapides et des plus complets que l'on puisse trouver sur un micro familial. Il est susceptible de pallier certaines lenteurs inhérentes aux langages de haut niveau et dispose d'effets graphiques et sonores surprenants.

3. Extension

La plupart des ordinateurs sont prévus pour recevoir des extensions matérielles telles qu'imprimantes, manettes de jeu, lecteurs de disquettes supplémentaires. Pourtant de nombreux micro-ordinateurs familiaux vous obligent à acheter des interfaces d'extension avant de pouvoir brancher un dispositif aussi simple qu'une imprimante ou une manette de jeu.

Bien que l'acheteur ne considère pas toujours ses besoins futurs, une machine possédant déjà une interface pour imprimante (type Centronics) et une interface pour manette de jeu est plus économique à long terme.

Le 6128 possède d'origine une interface Centronics, une entrée pour unité de disquette supplémentaire, une prise pour un lecteur de cassette (et une commande moteur), une interface pour deux manettes de jeu, une sortie stéréo et un bus d'extension pouvant recevoir des lecteurs de disquettes, des cartouches ROM, une interface série (AMSTRAD RS232C), des modems, des synthétiseurs de parole (AMSTRAD SSA2), des stylos optiques, etc...

4. Les effets sonores

Les caractéristiques sonores en micro-informatique vont du système qui fait des bruits plus ou moins cacophoniques à celui capable de simuler un instrument de musique électronique. Les 3 canaux et 8 octaves du 6128 lui permettent de produire des sons d'une bonne qualité musicale avec contrôle total de l'amplitude et des enveloppes de tonalité. De plus, l'émission est authentiquement stéréophonique, avec un canal à gauche, un à droite et un troisième au milieu.

Vous pourrez ainsi produire des ambiances sonores impressionnantes, synchronisées avec les effets graphiques.

Finalement, c'est à vous de voir ce qui est le plus important pour vous. Nous sommes sûrs que vous trouverez à votre machine des qualités que nous n'aurions même pas soupçonnées.

Pourquoi un micro ne peut-il pas faire cela ?

Les utilisateurs se demandent souvent pourquoi un ordinateur ne peut produire des images comme on en voit à la télévision. Pourquoi, par exemple, un micro ne peut-il animer un dessin comme dans un dessin animé ou un film ? Pourquoi les personnages sont-ils toujours simplifiés ?

La réponse est simple et complexe à la fois. La réponse simple est que l'écran d'un ordinateur est beaucoup plus sommaire qu'un écran TV (même si l'ordinateur utilise un poste TV, il n'utilise qu'une partie des circuits). La capacité mémoire d'une image retransmise à la télé est environ 20 fois plus importante que celle d'un ordinateur. De plus, pour être animés, tous les points de cette image doivent changer 50 fois par seconde. Les ordinateurs capables d'une telle performance sont encore de nos jours hors de portée de bien des budgets.

Les micros doivent se débrouiller avec une mémoire relativement petite comparée aux méga-ordinateurs. Leurs images sont donc plus saccadées et leur résolution d'écran bien moins élevée. Bien que l'on s'en approche un peu plus chaque jour, on est encore assez loin des dessins animés qu'on voit à la télévision.

Le clavier : un air de déjà vu...

Pourquoi ne peut-on simplement s'asseoir devant un micro et taper une page de texte dans la machine ?

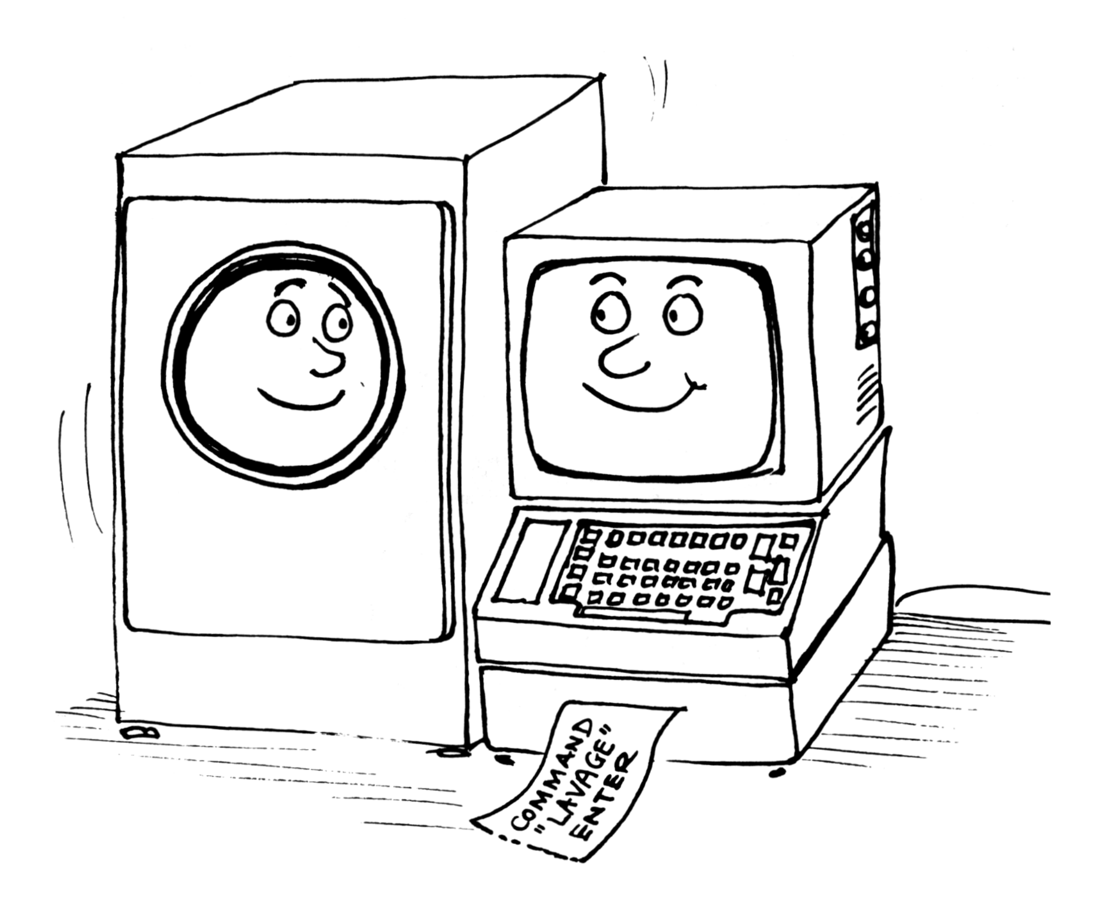
Ne vous laissez pas induire en erreur par le fait qu'un ordinateur ressemble à une machine à écrire avec un écran ; c'est un appareil qui communique avec vous par l'intermédiaire d'un langage de programmation. Et aussi, à moins que vous ne le dotiez d'un progiciel de traitement de texte, il vous répondra par le message :

```
Syntax error
```

...lorsque vous taperez n'importe quelle phrase suivie de [**RETURN**].

Pour taper une phrase puis **[RETURN]** et attendre de l'ordinateur qu'il se comporte comme une feuille de papier électronique dans une machine à écrire sophistiquée, il faut d'abord lui avoir chargé un programme lui disant : « nous allons taper une lettre ».

L'ordinateur semble combiner plusieurs appareils familiers : un écran de télévision, un clavier de machine à écrire et un magnétophone à cassette. Vous devez toutefois vous rappeler que ces ressemblances sont superficielles, l'ordinateur ayant des capacités d'un autre ordre et étant destiné à un usage complètement différent - une personnalité au-delà des apparences !



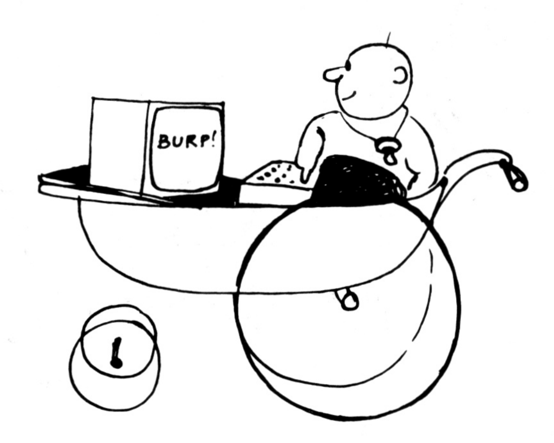
Mais qui a peur du jargon informatique ?

Comme beaucoup d'industries spécialisées, l'informatique a développé son propre jargon, un moyen rapide de parler de choses compliquées qui demanderaient beaucoup d'explications en langage courant. Tous les corps de métiers utilisent plus ou moins une forme de jargon leur permettant de parler de leur métier de la façon la plus concise possible.

Une des différences avec le jargon légal, par exemple, est que les confusions de ce dernier viennent plus de la façon dont les mots sont utilisés que des mots eux-mêmes.

Le jargon informatique utilise des mots simples, empruntés au langage courant (bien souvent anglophone), pour décrire directement et sans ambiguïté les concepts précis de la science qu'il sert. Au contraire d'une barrière, il est un outil de communication puissant, à l'image des langages de programmation, dont certains seraient tentés d'élever la maîtrise au rang d'expression artistique !

L'informatique est comprise rapidement par les jeunes qui en apprécient la précision et la simplicité. Ne voit-t-on pas plus de programmeurs de 10 ans que d'avocats du même âge ?



Bases du BASIC

Pratiquement tous les micro-ordinateurs familiaux utilisent le langage appelé BASIC. Il est maintenant passé au rang de langage performant destiné à la conception de programmes d'une puissance et d'une complexité extrême.

Son nom reste toutefois synonyme de facilité et guide les débutants de l'informatique personnelle du simple programme à des applications de statistiques élaborées.

Le BASIC est un langage interactif qui interprète un certain nombre de commandes bien définies puis effectue divers traitements sur les données qui lui sont fournies. Contrairement aux langages humains disposant de 5000 à 8000 mots, BASIC se débrouille avec environ 200 mots. Les programmes écrits en BASIC doivent obéir à des règles simples et rigides et respecter une syntaxe précise, implacablement sanctionnée par le message :

```
Syntax error
```

Ce n'est pas aussi contraignant qu'il le semblerait tout d'abord - un simple moyen d'indiquer que l'ordinateur ne comprend pas ce que vous voulez dire (bien souvent par la faute d'une virgule ou de guillemets mal placés, ou encore d'un O confondu à un 0 (zéro)). Les ordinateurs sont des machines ne demandant qu'à vous servir, pour peu que vous compreniez et respectiez leurs impératifs de fonctionnement. Ce sont avant tout des machines à calculer, pourvues, il est vrai, d'opérateurs sophistiqués dépassant le stade de la simple opération arithmétique, mais conservant encore la rigidité inhérente au traitement numérique.

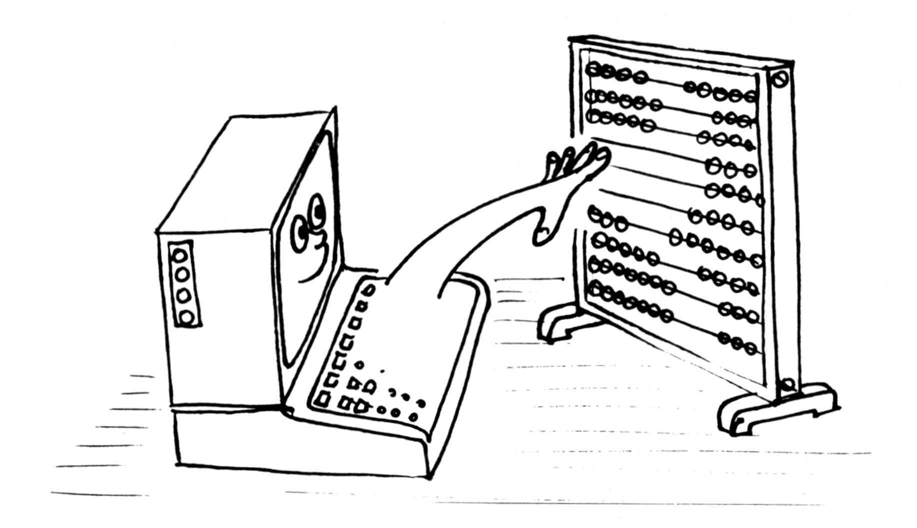
Des chiffres s'il vous plaît...

Si vous aviez les œuvres complètes de Victor Hugo sur un ordinateur, vous ne trouverez pas une seule lettre ou un seul mot dans le système : toute donnée, tout renseignement est d'abord transformé en chiffres et en nombres avant que l'ordinateur ne soit capable de les manipuler.

Le BASIC interprète les mots comme des chiffres afin que l'ordinateur puisse les traiter à l'aide d'additions, de soustractions ou d'opérations booléennes. Il est aussi à même de comparer ces données (tables de vérité, oui-non, etc.), de vérifier que deux conditions sont remplies en même temps (et, ou inclusif, ou exclusif), etc.

Grâce au programme, l'ordinateur divise toute tâche en une infinité d'opérations « oui-non ».

Un ordinateur ne sait même pas multiplier directement : pour multiplier 35 par 10, il ajoute 35 dix fois de suite.



Si ce procédé vous semble compliqué vous n'avez pas tout à fait tort. Vous venez même de découvrir une des vérités premières de la programmation. L'ordinateur est simplement une machine accomplissant des tâches répétitives extrêmement vite et avec une précision absolue. Le procédé qui consiste à traiter des données 0 ou 1 est souvent appelé numérique, par opposition au traitement analogique, jouant sur la variation continue du signal électronique.

Mais dans un monde réduit aux deux chiffres, 0 et 1, comment allons-nous compter ?

Bits et octets

Nous sommes habitués au système décimal, ou système à base 10, qui utilise 10 chiffres de 0 à 9. Le système qui utilise les 2 chiffres 0 et 1 s'appelle le système binaire et les unités qui le composent sont des BITS (pour Binary digITS).

7 = 00111 en binaire

Pour convertir un nombre décimal en nombre binaire (nombre en base 2) il faut partir des puissances de 2.

$2^0 = 1$
$2^1 = 2 = 2 = 2(2^0)$
$2^2 = 4 = 2x2 = 2(2^1)$
$2^3 = 8 = 2x2x2 = 2(2^2)$
$2^4 = 16 = 2x2x2x2 = 2(2^3)$

...et ainsi de suite :

2^4	2^3	2^2	2^1	2^0		
1	0	0	1	1		
(16 +	0 +	0 +	2 +	1)	=	19

Pour pouvoir utiliser les bits, on les groupe par ensembles de 8, que l'on appelle octets. Le plus grand nombre qu'on puisse écrire avec un octet est 11111111 (binaire) = 255 (décimal), soit 256 combinaisons (00000000 compris) de 0 et de 1.

Les ordinateurs manipulent les nombres par multiples de huit bits. 256 n'étant pas un nombre très grand, les manipulations sont effectuées au moyen de 2 octets disposés en matrice. Une adresse verticale et une adresse horizontale permettant de positionner chaque élément de la matrice.

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5				1		1				
6										
7										
8										
9										

Cette matrice peut placer (10x10) éléments, les éléments dotés de 1 étant reconnus comme 5,3 et 5,5.

Une matrice binaire 256x256 peut ainsi manipuler 65536 éléments différents en utilisant un octet pour l'adresse verticale et un octet pour l'adresse horizontale.

Pour raccourcir, on utilise le Kilo octet (K ou KO) qui est égal à 1024 octets (le kilo ne valant pas 1000 en informatique). Voilà pourquoi un ordinateur comme le CPC664, considéré comme un 64K dispose en fait de 65536 octets (64 x 1024) de mémoire.

Heureusement, le BASIC opère les conversions pour vous et vous pourrez devenir un très bon programmeur sans trop connaître le binaire. Cela vous aidera toutefois à comprendre pourquoi certains nombres ont plus d'importance que d'autres en informatique.

Cependant...

Bien que simple et élégant, le système binaire est difficile à lire car ses nombres ont tendance à s'allonger et à ne plus sauter aux yeux.

On a donc choisi de recourir à un système de notation en base 16, comme puissance de 2 capable de traduire les nombres binaires d'une manière plus concise. On l'appelle le système hexadécimal, HEX pour les intimes.

Décimal

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

HEX

0 1 2 3 4 5 6 7 8 9 A B C D E F

Le système hexadécimal peut décomposer les 8 bits d'un octet en deux blocs de quatre bits, 15 s'exprimant par un nombre à quatre bits : 1111 en binaire.

Considérons la table avec les notations binaire, décimal et hexadécimal :

Décimal	Binaire	Hexadécimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

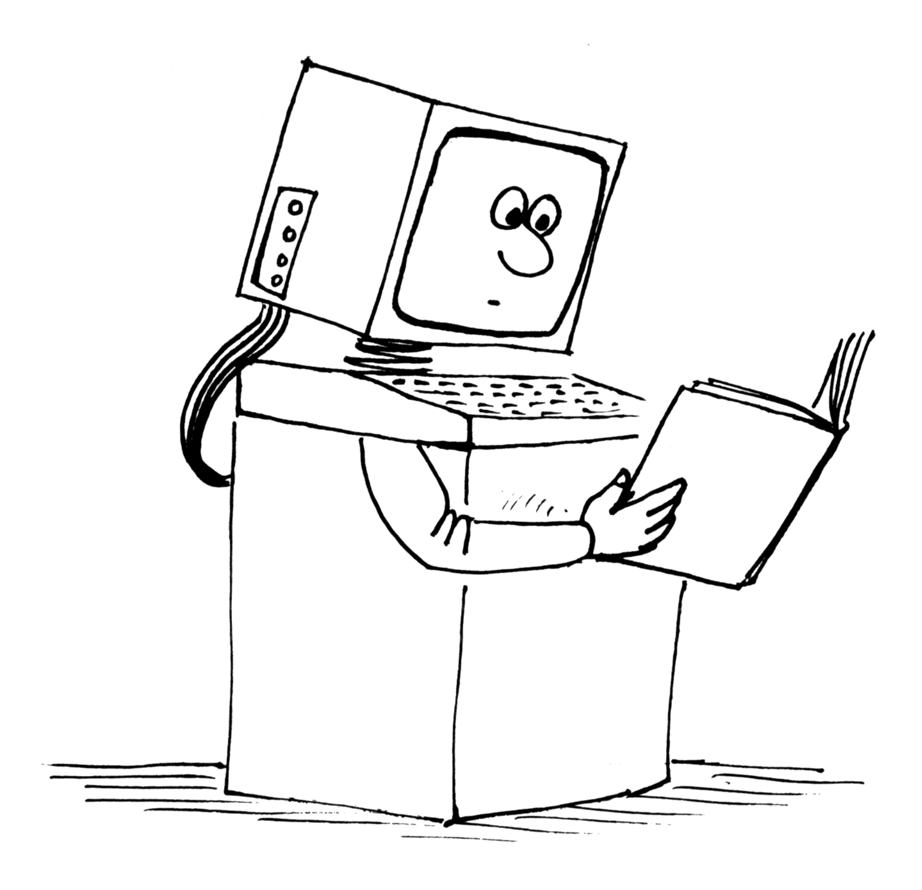
Un nombre à 8 bits peut être décomposé en deux nombres de 4 bits (appelés nibbles en anglais, ce qu'on peut traduire par petite bouchée...).

Dans ce livre, les nombres hexadécimaux sont désignés par le symbole &, &D6 par exemple, et servent aux programmeurs qui utilisent le langage assembleur, à mi-chemin entre le langage machine et le BASIC.

Pour convertir de l'hexadécimal, vous devez faire attention à multiplier le premier chiffre par 16 et lui ajouter le deuxième. Ainsi &D6 donne (13x16)+6=214 et non pas 13 et 6 ou 136, comme on serait tenté de le faire.

C'est le même procédé que celui du système décimal, à ceci près qu'il est plus facile de multiplier par 10 que par 16 !

Si vous avez tout compris jusque là, bravo, vous saurez rapidement maîtriser le CPC664 pour produire quantité de programmes. Si cela vous paraît compliqué, consultez un livre d'initiation qui vous expliquera plus en détail ce que nous ne pouvons qu'effleurer dans ce livre.



Partie 2 :
Fonctions propres au 6128

Cette partie approfondit certains aspects propres au 6128. Le Cours Elémentaire et le chapitre « Liste complète des mots clés du BASIC Schneider 6128 » vous fourniront toutes les informations de base dont vous pouvez avoir besoin.

Sujets abordés

* Jeu de caractères
* ASCII
* Variables
* Logique
* Caractères définis par l'utilisateur
* Formatage de PRINT
* Fenêtres
* Interruptions
* Données
* Sons
* Graphiques
* Matériel
* Graphiques exploitant la mémoire supplémentaire

Parlons un peu de caractère

Lorsque vous tapez sur le clavier de votre 6128, ne vous fiez pas aux lettres et nombres que vous voyez s'afficher à l'écran. Comme nous l'avons déjà dit, votre ordinateur n'est pas une machine à écrire. Ce qui apparaît n'est que le résultat d'une combinaison de contacts électriques déclenchés par la pression des touches. Les signaux électriques déclenchés par cette action sur les touches sont traduits, dans les circuits internes de l'ordinateur, afin de produire à l'écran certains motifs de points. Ces motifs forment des lettres, des nombres ou des symboles propres au jeu de caractères du 6128.

Certains caractères affichés ne sont pas directement accessibles par les touches du clavier mais peuvent être visualisés au moyen de l'instruction **PRINT CHR$** (<nombre>). Les éléments stockés dans l'ordinateur se présentent tous sous forme d'octets offrant, comme nous venons de le voir dans la partie 1 de ce chapitre, 256 combinaisons de valeurs possibles. L'ordinateur devant faire appel à au moins un octet pour chaque caractère stocké dans sa mémoire (que vous le vouliez ou non, c'est la plus petite unité gérée par le 6128) les 256 possibilités ainsi disponibles dépassent largement les 96 caractères, dits standard, d'une machine à écrire, ménageant ainsi 160 octets supplémentaires dont il serait stupide de ne pas tirer parti.

L'ensemble des caractères « standard » s'appelle un « sous-jeu ». Dans le monde de l'informatique, il s'inscrit dans la catégorie du système d'affichage « ASCII », acronyme d'American Standard Code for Information Interchange. A l'origine, ce système assurait la transmission de données d'un ordinateur à l'autre sous une forme que les systèmes reconnaissent. Vous trouverez dans le chapitre « Pour information... » la liste des caractères ASCII ainsi que les caractères supplémentaires du 6128 accompagnés des codes numériques correspondant.

Comment en arriver là...

Au point où nous en sommes, le programme d'affichage du jeu de caractères :

```
10 FOR n=32 to 255
20 PRINT CHR$(n);
30 NEXT
```

...n'a certainement plus de secrets pour vous. Décortiquons-le maintenant :

Nous constatons tout d'abord qu'au lieu d'ordonner **PRINT** « **abcdefghijklmn**...etc », nous avons entré **PRINT CHR$(n)**, **n** étant une représentation pratique de « variable ». Une variable est un élément d'information qui « varie » selon les instructions du programme. (Le choix de **n** pour variable est tout à fait arbitraire, nous aurions pu prendre n'importe quelle combinaison de lettres, pourvu qu'elle soit différente d'un mot clé).

Comment reconnaître une variable ?

Le chiffre 5 est fixe, il est compris entre 4 et 6 ; il n'est donc pas une variable. Le caractère n est également fixe, c'est une lettre de l'alphabet.

Comment l'ordinateur peut-il donc faire la différence ? Si la lettre **n** avait été déclarée comme caractère alphabétique, elle aurait été tapée entre guillemets **" n "**, et l'ordinateur, incapable de comprendre la séquence **FOR " n " = 32 TO 255**, aurait répondu par le message **Syntax error**.

En tapant **n** sans guillemets, nous avons indiqué à l'ordinateur qu'il s'agissait d'une variable. En BASIC, une commande **FOR** doit être suivie d'une variable ; l'ordinateur rejette donc tout que ce qui en diffère.

Nous venons d'indiquer à l'ordinateur que **n = 32 TO 255**, définissant ainsi les valeurs possibles pour la variable. En fait, il s'agit d'une séquence commençant à **32** et finissant à **255**.

Après avoir déclaré cette variable, nous devons alors indiquer à l'ordinateur ce qu'il doit en faire ; c'est justement le rôle de la ligne **2Ø** :

```
2Ø PRINT CHR$(n);
```

Quelle que soit la valeur de **n**, l'ordinateur doit rechercher dans sa mémoire le caractère correspondant, afin de l'afficher à l'écran.

Le point-virgule terminant la ligne **2Ø** indique à l'ordinateur qu'il ne doit effectuer ni retour chariot ni de saut de ligne. (Car chaque caractère serait affiché en première colonne d'une nouvelle ligne.)

La ligne **3Ø** lui commande de revenir à la ligne contenant **FOR**, après avoir opéré sur la première valeur de **n** (à savoir **32**), et d'effectuer la même opération pour la valeur suivante (**NEXT**) de la variable **n**. Ce procédé, que l'on appelle « boucle », est l'un des concepts primordiaux de la programmation. Il vous évite de retaper de longues séquences d'instructions et vous apprendrez très vite à vous en servir dans vos programmes.

Lorsque la boucle **FOR NEXT** atteint la dernière valeur de la fourchette déclarée (**255**), cette opération s'arrête et l'ordinateur cherche la ligne suivant la ligne **3Ø**. Comme il n'y en a pas, le programme cesse et revient au mode direct avec affichage du message **Ready**, signalant ainsi qu'il est prêt à recevoir de nouvelles instructions. Vous pouvez alors, si vous le désirez, entrer **RUN** une nouvelle fois pour répéter l'exécution du programme. Ce dernier est stocké dans une partie de la mémoire où il restera tant que vous ne donnez pas l'ordre de le retirer et tant que l'ordinateur reste sous tension.

Ce programme est l'illustration parfaite d'un aspect fondamental de l'informatique : tout ce que fait l'ordinateur se rapporte à des nombres. Il a affiché l'alphabet et tout un ensemble de caractères par le biais de leur référence numérique. Lorsque vous appuyez sur la touche **A**, vous ne demandez pas l'affichage du **A** mais vous invitez l'ordinateur à rechercher dans sa mémoire l'information numérique correspondant à l'affichage du **A** sur l'écran. L'emplacement de cette information est défini par un code numérique mis en œuvre par l'actionnement de la touche du clavier.

Chaque caractère correspond à un nombre. Vous trouverez une liste de références dans la partie 3 du chapitre « Pour information... ».

De même, l'affichage des caractères à l'écran n'a rien à voir avec l'écriture telle que vous la connaissez : là encore, il s'agit de nombres.

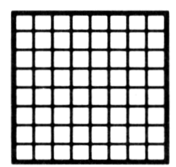
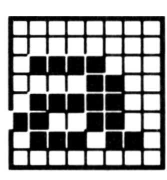
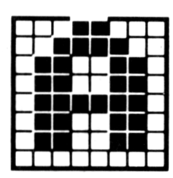
Ainsi, en code ASCII la lettre A correspond à 97. Mais l'ordinateur ne comprend pas ce nombre (il n'est pas très futé) ; 97 doit d'abord être converti du code décimal en un code que l'ordinateur puisse reconnaître, c'est-à-dire en code (langage ???) machine (dont les principes sous-jacents sont abordés précédemment dans ce chapitre).

A première vue, la conversion des nombres décimaux en notation hexadécimale peut sembler laborieuse. Nous sommes tellement habitués à compter en base 10, qu'en changer nous paraît aussi farfelu que d'inverser la disposition du couteau et de la fourchette sur une table !

L'assimilation de la notation hexadécimale exige une petite gymnastique intellectuelle qui, après que vous l'aurez maîtrisée, vous ouvrira la porte sur un grand nombre de notions informatiques.

Si les systèmes de numération binaire et héxadécimale restent flous pour vous, nous vous conseillons de lire ou de relire attentivement la première partie de ce chapitre.

Une fois que l'ordinateur a traduit l'activation de la touche A par une série de nombres qu'il comprend, il recherche la partie de la mémoire indiquée afin d'obtenir une autre série de nombres définissant à leur tour le caractère à l'écran. La lettre que vous voyez apparaître est donc constituée d'un bloc de données stocké en mémoire sous forme de « matrice » numérique :

MATRICE GRILLE a MINUSCULE A MAJUSCULE

La matrice se compose de lignes et de colonnes de points. Le caractère s'affiche par activation ou désactivation de la séquence de points requise, chaque point étant déterminé par des données stockées dans le mémoire de l'ordinateur. Dans le cas du 6128, les matrices ou « cellules » de l'écran sont constituées de 8 lignes sur 8 colonnes. Si le caractère que vous désirez afficher n'existe pas parmi les **255** disponibles, vous pouvez le définir à l'aide du mot clé **SYMBOL**, décrit dans une partie ultérieure de ce chapitre.

Les « caractères définis par l'utilisateur » s'obtiennent en combinant chacun des 64 points de la matrice, ce qui représente un grand nombre de possibilités. De plus, la capacité de réunir des blocs de caractères pour constituer des caractères plus larges ouvre les possibilités graphiques à votre seule disponibilité ou imagination.

Expressions logiques

Une des grandes différences entre un simple calculateur et un ordinateur réside dans la capacité qu'a l'ordinateur à manipuler les opérations logiques d'applications telles que la séquence **IF**...**THEN** (si...alors). Les opérateurs logiques traitent les valeurs auxquelles ils sont appliqués en tant qu'ensembles de bits sur lesquels ils opèrent. Leur description et leur utilisation est tout à fait ...ben, logique, mais il est très difficile de décrire la logique en termes simples.

Disons qu'une expression logique comprend deux arguments et un opérateur logique. Une expression logique comprend :

<argument>[<opérateur logique> <argument>]

où l'argument est défini comme suit :

<argument> = **NOT** <argument>
<expression numérique>
<expression relationnelle>
<expression logique>

Les deux arguments de part et d'autre d'un opérateur logique sont convertis en nombres entiers (ce qui peut dans certains cas provoquer l'erreur N° 6, lorsque les nombres à convertir n'appartiennent pas à l'intervalle des nombres entiers de l'ordinateur).

Les opérateurs logiques sont, par ordre de priorité :

AND (=et) résultat 0 sauf si les deux arguments sont vrais (bits indicateurs tous les deux 1)
OR (= ou inclusif) résultat 1 sauf si les deux arguments sont faux (bits indicateurs tous les deux 0)
XOR (= ou exclusif) résultat 1 sauf si les deux arguments sont identiques (bits indicateurs identiques).

AND est le plus courant et ne veut pas dire « ajouter ».

```
PRINT 10 AND 10
```

Résultat **10** ! Mais

```
PRINT 10 AND 12
```

Résultat **8** et

```
PRINT 10 AND 1000
```

Résultat **8** aussi !

Ceci est dû au fait que **10** et **1000** ont été convertis en leur équivalent binaire. Drôles de mathématiques, mais rappelez-vous que les opérateurs logiques ne connaissent que le système binaire 0 et 1, vrai ou faux.

Explication :

10 en binaire	1010
1000 en binaire	1111101000

L'opérateur **AND** vérifie les **1** se trouvant en même position dans les deux nombres et trouve :

0000001000

...qui en système décimal est égal à **8**. L'opérateur logique **AND** (=et) vérifie si deux conditions sont vraies en même temps. Voici un petit programme explicatif :

```
10 INPUT "Aujourd'hui nous sommes le";jour
20 INPUT "Quel est le numero du mois";mois
30 IF jour=25 AND mois=12 THEN 50
40 CLS:GOTO 10
50 PRINT"Joyeux Noel !"
```

OR opère également par bits, donnant **1** dans tous les cas où les deux bits des arguments diffèrent de **0**, auquel cas le résultat est égal à **0**. Avec les mêmes nombres que dans l'exemple **AND** :

```
PRINT 1000 OR 10
1002
```

En binaire : 1010
1111101000

Résultat : 1111101010

Dans un programme :

```
10 CLS
20 INPUT "Quel est le numero du mois";mois
30 IF mois=12 OR mois=1 OR mois=2 THEN 50
40 GOTO 10
50 PRINT"Nous devrions etre en Hiver !"
```

L'opérateur **NOT** inverse chaque bit de l'argument (0 devient 1 et inversement) :

```
10 CLS
20 INPUT "Quel est le numero du mois";mois
30 IF NOT (mois=6 OR mois=7 OR mois=8)THEN 50
40 GOTO 10
50 PRINT"Nous ne pouvons etre en Ete !"
```

On peut aussi mélanger autant d'opérateurs logiques qu'on le désire (dans la limite d'une ligne) pour donner, par exemple :

```
10 INPUT "Aujourd'hui nous sommes le";jour
20 INPUT "Quel est le numero du mois";mois
30 IF NOT ( mois=12 OR mois=1) AND jour=29 THEN 50
40 CLS:GOTO 10
50 PRINT"Ce n'est ni Decembre ni Janvier  mais  c'est peut-
etre une annee bissextile"
```

Le résultat d'une expression relationnelle est soit **-1**, soit **Ø**. La représentation binaire de **-1** donne une configuration binaire dont tous les bit valent **1**, et celle de **Ø** en donne une dont tous les bits valent **Ø**. Le résultat d'une opération logique sur de tels arguments donnera **-1** pour vrai et **Ø** pour faux.

Ajoutons deux lignes pour compliquer :

```
60 PRINT NOT(mois=12 OR mois=1)
70 PRINT (mois=12 OR mois=1)
```

Si vous lancez le programme avec **29** pour le jour et **2** pour le mois, vous obtenez la réponse de la ligne **5Ø** et les valeurs produites par les expressions des lignes **6Ø** et **7Ø**.

Pour finir, **XOR** (OU eXclusif) donne un résultat vrai si les deux arguments sont différents.

Résumons par un tableau susceptible de donner une idée plus claire de la situation. Ce type de représentation est appelé table de vérité d'une fonction.

ARGUMENT A	1010
ARGUMENT B	0110
Résultat AND (et)	0010
Résultat OR (ou)	1110
Résultat XOR (ou exclusif)	1100

Caractères définis par l'utilisateur

Une des premières applications de nombres binaires que vous êtes susceptibles de rencontrer concerne la détermination de caractères à l'aide de la commande **SYMBOL**. Si le caractère est dessiné sur une grille de 8 sur 8, chaque case de la grille est alors convertie en chiffre binaire ; 1 correspondant à une case encrée et 0 à une case sans encre, c'est-à-dire de la couleur du fond. Les huit nombres sont ensuite passés comme paramètres de la commande **SYMBOL**. Pour dessiner une maison, par exemple :

```
    *     = 00001000 = &08 =               8        = 8
   ****   = 00111100 = &3C =           32+16+8+4    = 60
  *    *  = 01000010 = &42 =        64          +2  = 66
 * *  * * = 10100101 = &A5 = 128    +32      +4  +1= 165
 *      * = 10000001 = &81 = 128                 +1= 129
 * ** * * = 10110101 = &B5 = 128    +32+16   +4  +1= 181
 * **   * = 10110001 = &B1 = 128    +32+16       +1= 177
 ******** = 11111111 = &FF = 128+64+32+16+8+4+2+1= 255
```

...la commande est celle-ci :

```
SYMBOL 240,8,60,66,165,129,181,177,255
```

...ou :

```
SYMBOL 240,&08,&3C,&42,&A5,&81,&B5,&B1,&FF
```

...ou bien :

```
SYMBOL 240,&X00001000,&X00111100,&X01000010,&X10100101
           &X10000001,&X10110101,&X10110001,&X11111111
```

Pour afficher le caractère défini, vous devez entrer :

```
PRINT CHR$(240)
```

Et enfin, pour grouper plusieurs blocs de caractères il vous suffira de spécifier :

```
semi$=CHR$(240)+CHR$(240)
PRINT semi$
```

...ou :

```
terrasse$=string$(15,240)
PRINT terrasse$
```

Lancez les rotatives !

PRINT est une des premières commandes que vous utilisez lorsque vous débutez en informatique. Elle fait partie des commandes du **BASIC** dont le nom indique réellement la fonction...en apparence. En fait, **PRINT** suppose beaucoup plus de choses qu'il n'en paraît : où placer l'affichage ? De quelle manière ?

Format de PRINT

La commande **PRINT** peut s'utiliser de plusieurs façons, la manière la plus simple étant de la faire suivre de l'élément à afficher ou imprimer : un nombre, une chaîne ou un nom de variable.

```
PRINT 3
 3

PRINT "bonjour"
bonjour

a=5
PRINT a
 5

a$="test"
PRINT a$
test
```

L'instruction **PRINT** peut contenir plusieurs éléments intercalés d'un séparateur, **TAB** ou **SPC**. Les virgules et les points-virgules sont des séparateurs. Le point-virgule permet l'affichage aligné des éléments et la virgule provoque le décalage de l'affichage sur la zone suivante. Bien que la largeur initiale d'une zone soit de treize caractères, vous pouvez la modifier à l'aide de la commande **ZONE**.

```
PRINT 3;-4;5
 3 -4  5

PRINT "bonjour ";"Monsieur"
bonjour Monsieur

PRINT "bonjour ","Monsieur"
bonjour              Monsieur


PRINT 3,-4,5
 3            -4            5

ZONE 4
PRINT 3,-4,5
 3  -4   5
```

Les nombres positifs sont précédés d'un espace et les nombre négatifs du signe moins. Il sont tous suivis d'un espace. Les chaînes s'affichent exactement telles qu'elles apparaissent dans les guillemets.

La fonction **SPC** accepte un paramètre numérique définissant le nombre d'espaces à afficher. Pour les valeurs négatives le nombre pris en compte est zéro, si la valeur est supérieure à la largeur de la fenêtre, c'est la largeur de la fenêtre qui est alors prise en compte.

```
PRINT SPC(5) "coucou"
     coucou


x=3
PRINT SPC(x*3)"coucou"
         coucou
```

TAB fonctionne de manière identique, mais affiche le nombre d'espaces requis de façon à ce que l'élément à afficher apparaisse à la colonne spécifiée.

Le canal où apparaissent toutes les données en sortie est la fenêtre 0, à moins qu'un spécificateur de canal (#) ne précède la liste d'éléments à afficher. Vous pouvez utilisez d'autres canaux pour la sortie sur d'autres fenêtres. Les canaux 8 et 9 sont particuliers : le canal 8 correspond à une imprimante et le canal 9 achemine les données sur un fichier de disquette (ou de cassette). Dans ces cas-là, la commande **WRITE** est toutefois préférable à **PRINT**.

`PRINT "bonjour"` `bonjour`	- fenêtre 0
`PRINT #0,"bonjour"` `bonjour`	- fenêtre 0 aussi
`PRINT #4,"bonjour"` `bonjour`	- fenêtre 4 (En haut de l'écran)
`PRINT #8,"bonjour"` `bonjour`	- sur l'imprimante (Le cas échéant)

Bien que **TAB** et **SPC** conviennent parfaitement aux formats d'affichage simples, une visualisation plus détaillée des informations demandera l'utilisation de la commande **PRINT USING**, accompagnée du modèle de format adéquat. Un modèle de format est une chaîne contenant des caractères spéciaux, spécifiant chacun un type de format particulier. Ces caractères, appelés « spécificateurs de format de zone » font l'objet d'une approche plus détaillée dans la description du mot clé **PRINT USING**, (précédemment dans cet ouvrage). Les quelques exemples suivants vous apporteront cependant des éclaircissements sur le sujet.

Pour commencer, voici les formats destinés à l'affichage des chaînes :

```
PRINT USING "\        \";"test de chaine"
test d
```

" ! " sert à afficher le premier caractère d'une chaîne.

```
PRINT USING "!";"test de chaine"
t
```

Le format le plus utile pour les chaînes est encore " **&** ". Il empêche l'affichage d'une chaîne sur une nouvelle ligne. En effet, lorsqu'une chaîne ne tient pas sur la ligne en cours, le BASIC la reporte par défaut sur une nouvelle ligne. **PRINT USING " & "**; permet de passer outre.

(Utilisez **BORDER Ø** pour visualiser les bords du papier.)

```
MODE 1:LOCATE 39,1:PRINT "trop longue"

                                                -ligne 1
trop longue                                     -ligne 2

mode 1:locate 39,1:PRINT USING "&";"trop longue"

                                   tro          -ligne 1
p longue                                        -ligne 2
```

Pour l'affichage de nombres, vous disposez de multiples modèles. Le plus simple étant PRINT USING ##### , où chaque # correspond à un chiffre.

```
PRINT USING "######";123
   123
```

Vous pouvez introduire le point décimal à l'aide de " . ".

```
PRINT USING "####.#####";12.45
  12.45000
```

Les chiffres avant le point décimal peuvent être regroupés par trois et séparés par des virgules. Il faut alors faire appel à " , " dans le modèle avant le point décimal.

```
PRINT USING "########,.####";123456.78
  123,456.7800
```

« **$$** » et « **FF** » permettent d'inclure des signes de monnaie flottants affichés avant le premier chiffre du nombre même s'il ne remplit pas le format entier.

```
PRINT USING "$$##";7
  $7

PRINT USING "$$##";351
$351

PRINT USING "F####,.##";1234.567
F1,234.57
```

Vous remarquez que le résultat est arrondi.

" ** " dans le modèle remplace les blancs précédant le nombre par des astérisques.

```
PRINT USING "******.#";12.22
****12.2
```

Vous pouvez combiner les astérisques aux symboles monétaires : " ****$**... etc " ou " ****F**,... etc ".

Le signe " + " au début du modèle de format demande le placement du signe du résultat, avant le premier chiffre. S'il figure à la fin du modèle, le signe est placé après le résultat.

Le signe " - " ne peut se trouver qu'à la fin du modèle de format pour qu'apparaisse un signe moins après les nombres négatifs.

```
PRINT USING "+##";12
+12

PRINT USING "+##";-12
-12

PRINT USING "##+";12
12+

PRINT USING "##-";-12
12-

PRINT USING "##-";12
12
```

Pour exprimer un nombre sous forme exponentielle, vous pouvez utiliser « ↑ ↑ ↑ ↑ » dans le modèle.

```
PRINT USING "###.##↑↑↑↑";123.45
 12.35E+01
```

Lorsque vous utilisez des modèles de format, le symbole % apparaît devant le résultat si le nombre dépasse le modèle spécifié (afin d'indiquer que le résultat n'a pas été raccourci pour satisfaire au modèle).

```
PRINT USING "####";123456
%123456
```

A vos fenêtres !

Le BASIC du 6128 permet de définir jusqu'à huit fenêtres de texte. Toutes les commandes d'acheminement du texte vers l'écran peut servir à envoyer l'affichage sur l'une de ces fenêtres.

La commande de définition d'une fenêtre s'intitule tout naturellement **WINDOW**. Elle est suivie de 5 valeurs. La première, facultative, spécifie la fenêtre désirée (la fenêtre Ø par défaut). Tous les messages du BASIC (Ready par exemple) s'affichent dans la fenêtre Ø. Précédant ce chiffre, le symbole dièse (#) l'identifie en tant que spécificateur de canal. Les quatre paramètres restant correspondent aux limites gauche, droite, supérieure et inférieure de la fenêtre. Ces valeurs définissent les limites en termes de lignes et colonnes d'écran et doivent donc être comprises entre 1 et 80 dans le sens horizontal et entre 1 et 25 pour la verticale.

L'exemple suivant définit une fenêtre 4 allant de la colonne 7 à la colonne 31, et de la ligne 6 à la ligne 18. Réinitialisez votre ordinateur et tapez :

```
WINDOW #4,7,31,6,18 '
```

Rien ne s'affiche après l'entrée de cette commande, mais si vous tapez :

```
INK 3,9
PAPER #4,3
CLS #4
```

...un grand rectangle vert apparaît à l'écran définissant la fenêtre 4. L'exemple ci-dessus montre que introduction du numéro de canal permet d'utiliser **PAPER** et **CLS** avec chacune des huit fenêtres. La commande agit par défaut sur la fenêtre Ø.

Toutes les commandes ci-dessous peuvent être accompagnées d'un spécificateur de canal identifiant la fenêtre sur laquelle elles vont agir.

```
CLS
COPYCHR$
INPUT
LINE INPUT
LIST
LOCATE
PAPER
PEN
POS
PRINT
TAG
TAGOFF
VPOS
WINDOW
WRITE
```

La fenêtre verte que vous venez de créer a obscurci une partie du texte déjà affichée (sur la fenêtreØ).

Le texte peut être dirigé vers une fenêtre donnée en intégrant un spécificateur de canal dans la commande **PRINT**.

```
PRINT #4,"coucou"
```

Ces mots apparaîtront en haut du rectangle vert au lieu de s'afficher sur la ligne suivante comme lors de l'entrée de :

```
PRINT "coucou"
```

Vous constatez qu'une partie de la fenêtre verte a été recouverte par le texte.

Si vous désirez voir tous les messages du BASIC s'afficher dans la fenêtre 4, par exemple, la commande **WINDOW SWAP** permet de la substituer à la fenêtre par défaut :

```
WINDOW SWAP 0,4
```

Le message **Ready** qui va suivre cette commande s'affiche alors dans la fenêtre verte. Le curseur se trouve immédiatement dessous. Tapez maintenant la ligne :

```
PRINT #4,"coucou"
```

...et le mot « coucou » va apparaître directement sous la commande **WINDOW SWAP** dans l'ancienne fenêtre **Ø**, (la fenêtre 4 à présent). Il ressort que la position d'un affichage à l'intérieur d'une fenêtre est stockée et que, même après un **WINDOW SWAP**, le texte s'affiche sur cette position plutôt qu'en haut. Entrez :

```
LOCATE #4,20,1
PRINT "la fenetre 0 est ici"
PRINT #4,"ceci est la fenetre 4"
```

Le message « **la fenêtre Ø est ici** » apparaît sur la ligne **PRINT** tandis que le message « **ceci est la fenêtre 4** » s'affiche au milieu de la première ligne de l'écran.

Avant l'entrée d'une commande **WINDOW**, les huit fenêtres couvrent la totalité de l'écran. C'est également vrai pour une commande **MODE**. Ainsi, lorsqu'après avoir utilisé des fenêtres, le curseur se trouve dans une fenêtre très réduite, il vous suffit de taper **MODE 1**, comme ceci :

```
MODE 1
WINDOW 20,21,7,18
MO
DE
```

Ne vous inquiétez pas si le mot **MODE** est coupé en deux, cela ne l'empêche pas de remplir sa fonction. Par contre, n'oubliez pas de laisser un espace entre **MODE** et **1**.

Vous connaissez maintenant le fonctionnement des fenêtres, essayez donc d'entrer ce petit programme :

```
10 MODE 0
20 FOR n=0 TO 7
30 WINDOW #n,n+1,n+6,n+1,n+6
40 PAPER #n,n+4
50 CLS #n
60 FOR c=1 TO 200:NEXT c
70 NEXT n
```

Il définit huit fenêtres de couleurs différentes se chevauchant. Lorsque l'exécution du programme est terminée et que le message « **Ready** » apparaît, appuyez sur **ENTER** plusieurs fois et observez le changement de couleur de l'écran que provoque le défilement de la fenêtre 0. Les blocs de couleur défilent, mais l'emplacement des autres fenêtres ne change pas. Entrez maintenant :

```
CLS #4
```

...pour constater que la quatrième fenêtre est toujours à la même place. Comme vous vous en doutiez, le nouveau bloc de couleur a obscurci ceux qu'il recouvre. A titre informatif, observez la différence entre les diverses entrées :

```
LIST
LIST #4
LIST #3
```

La commande **WINDOW** présente une autre particularité qu'illustre le dernier programme de ce chapitre : l'ordre d'entrée des dimensions gauche et droite des fenêtres importe peu. En effet, si le premier paramètre est supérieur au second, le BASIC trie automatiquement les dimensions et les replace dans l'ordre. Cette remarque est également valable pour les limites inférieures et supérieures des fenêtres.

```
10 MODE 0
20 a=1+RND*19:b=1+RND*19
30 c=1+RND*24:d=1+RND*24
40 e=RND*15
50 WINDOW a,b,c,d
60 PAPER e:CLS
70 GOTO 20
```

Puis-je vous interrompre ?

Vous l'avez peut-être remarqué, l'une des innovations logicielles majeures des ordinateurs Schneider consiste à gérer les interruptions à partir du BASIC : en d'autres termes, le BASIC Schneider est capable d'exécuter un certain nombre de tâches indépendantes à l'intérieur d'un même programme. Cette capacité, connue sous le terme de « multitâche », est gérée par les commandes **AFTER** et **EVERY**.

La possibilité de mettre les sons en file d'attente et de les synchroniser par des rendez-vous représente un autre exemple de cette particularité.

La synchronisation du système incombe à son horloge maîtresse. Elle est constituée d'un système de synchronisation à quartz, intégré dans l'ordinateur veillant à l'organisation chronologique des événements (le balayage de l'affichage et la fréquence du processeur, par exemple). Toutes les fonctions matérielles se rapportant au temps peuvent se retrouver dans l'horloge à quartz du système.

L'implémentation logicielle de cette synchronisation appartient aux commandes **AFTER** et **EVERY**. Fidèles à la convivialité du BASIC Schneider, celles-ci font exactement ce qu'elles indiquent : **AFTER** (après) la durée définie dans la commande, le programme donne le contrôle au sous-programme indiqué qui exécute la tâche définie.

Le 6128 gère en permanence une horloge temps réel. La commande **AFTER** permet à un programme en BASIC d'appeler des sous-programmes à un moment donné. Quatre chronomètres sont ainsi disponibles, chacun pouvant être associé à un sous-programme.

Après écoulement du délai spécifié, le sous-programme est automatiquement appelé, comme il le serait par une instruction **GOSUB**. A la fin du sous-programme, la commande **RETURN** provoque la reprise du programme principal à l'endroit même de l'interruption.

La commande **EVERY** permet à un programme en BASIC de rappeler des sous-programmes à intervalles réguliers. Cette fois encore, quatre chronomètres sont disponibles et chacun peut être associé à un sous-programme.

Les chronomètres possèdent différentes priorités d'interruption. Le chronomètre 3 correspond à la priorité supérieure et le chronomètre 0 à la priorité inférieure. (Pour de plus amples renseignements, consultez le chapitre « Pour information... »).

```
10 MODE 1:n=14:x=RND*400
20 AFTER x,3 GOSUB 80
30 EVERY 25,2 GOSUB 160
40 EVERY 10,1 GOSUB 170
50 PRINT"Testez vos reflexes"
60 PRINT"Appuyez sur la barre espace."
70 IF flag=1 THEN END ELSE 70
80 z=REMAIN(2)
90 IF INKEY(47)=-1 THEN 110
100 SOUND 1,900:PRINT"Tricheur !":GOTO 150
110 SOUND 129,20:PRINT"MAINTENANT":t=TIME
120 IF INKEY(47)=-1 THEN 120
130 PRINT"vous avez mis";
140 PRINT (TIME-t)/300;"seconde"
150 CLEAR INPUT:flag=1:RETURN
160 SOUND 1,0,50:PRINT".";:RETURN
170 n=n+1:IF n>26 THEN n=14
180 INK 1,n:RETURN
```

Les commandes **AFTER** et **EVERY** peuvent intervenir à tout moment pour réinitialiser le sous-programme concerné en accord avec le chronomètre. Ces deux commandes se partagent les chronomètres, une commande **AFTER** annulant toute commande **EVERY** antérieure pour un chronomètre donné et inversement.

Les commandes **DI** et **EI** désactivent et activent les interruptions de chronomètre tout en permettant l'exécution des commandes les séparant. Une interruption de priorité élevée est ainsi dans l'impossibilité de perturber le déroulement d'une interruption de priorité moindre. La fonction **REMAIN** désactive le comptage d'un des chronomètres et restitue le nombre d'impulsions restantes avant le déclenchement de l'interruption.

Utilisation des données...

Dans un programme nécessitant l'entrée d'un même ensemble d'informations dès le début, il serait judicieux de pouvoir entrer toutes les valeurs une fois pour toutes. C'est le rôle des commandes **READ** et **DATA**. **READ** est très similaire à **INPUT** dans le sens où elle affecte des valeurs à des variables. Elle en diffère cependant par le fait que ces valeurs sont lues à partir des commandes **DATA** au lieu d'être entrées au clavier. En voici deux exemples :

```
10 READ a,b,c
20 PRINT"les nombres sont";a;"et";b;"et";c
30 DATA 12,14,21
run

10 INPUT "Entrez 3 nombres separes par une virgule";a,b,c
20 PRINT"les nombres sont";a;"et";b;"et";c
run
```

Des virgules séparent les différents éléments d'une commande **DATA**, tout comme pour la commande **INPUT**.

Outre des valeurs numériques, les commandes **DATA** peuvent contenir des constantes alphanumériques :

```
10 DIM a$(10)
20 FOR i=0 TO 9
30 READ a$(i)
40 NEXT
50 FOR i=0 TO 9
60 PRINT a$(i);" ";
70 NEXT
80 DATA rien,ne,sert,de,courrir,il,faut,partir,a,point
run
```

Bien que la commande **DATA** comporte des chaînes, celles-ci ne se trouvent pas entre guillemets. L'emploi des guillemets est en effet facultatif pour la commande **DATA**, comme lors de l'entrée d'une chaîne en réponse à une commande **INPUT**. Ils sont cependant très utiles lorsque les chaînes de données contiennent elles-mêmes des virgules. En cas d'omission, l'instruction **READ** interprèterait ces virgules comme les virgules conventionnelles de la commande **DATA**, coupant ainsi la chaîne en plusieurs tronçons.

```
10 READ a$
20 WHILE a$<>"*"
30 PRINT a$
40 READ a$
50 WEND
60 DATA J'ai longtemps, habite,sous de vastes portiques
70 DATA Que, des, soleils marins, teignaient, de mille feux
80 DATA *
run
```

La chaîne de la ligne **60** comporte des virgules entraînant le lecture et l'affichage de chaque partie. Par contre, la chaîne de la ligne **70** est délimitée par des guillemets, et sera donc affichée comme un tout.

L'exemple ci-dessus montre que les données peuvent occuper plusieurs lignes. **READ** traite les lignes de haut en bas dans l'ordre des numéros de ligne (**60**, **70**, **80**, etc...). Par ailleurs, les instructions **DATA** peuvent se trouver n'importe où dans le programme : avant ou après la commande **READ** qui extrait les informations.

Si un programme contient plusieurs commandes **READ**, la seconde reprend là où la première s'arrête :

```
10 DATA 123,456,789,321,654,2343
20 FOR i=1 TO 5
30 READ nomb
40 total=total+nomb
50 NEXT
60 READ total2
70 IF total=total2 THEN PRINT"les donnees sont justes" ELSE
PRINT"il y a une erreur dans les donnees"
run
```

Essayez de modifier la ligne **10** en introduisant une erreur dans l'un des cinq premiers nombres, puis lancez le programme. Cette méthode d'insertion, à la fin d'une commande **DATA**, d'une valeur supplémentaire égale à la somme de toutes les autres valeurs constitue un bon moyen de détection d'éventuelles erreurs, surtout lorsque les lignes **DATA** sont nombreuses. Cette méthode s'appelle somme de contrôle (checksum).

Si un programme fait appel à des données mixtes (chaînes et nombres), il est possible de les combiner dans les instructions **DATA** et **READ**, à condition que les éléments soient lus correctement. Par exemple, si **DATA** contenait des séquences de deux nombres suivis d'une chaîne, seul l'emploi d'une commande **READ** suivie de deux variables numériques et d'une chaîne serait justifié :

```
10 DIM a(5),b(5),p$(5)
20 FOR i=1 TO 5
30 READ a(i),b(i),p$(i)
40 NEXT
50 DATA 1,7,fred,3,9,francois,2,2,dany,4,6,olivier,9,1,eric
60 FOR i=1 TO 5
70 PRINT p$(i),":";a(i)*b(i)
80 NEXT
```

Vous pouvez, pour changer, séparer les différents types de données :

```
10 DIM a(5),b(5),p$(5)
20 FOR i=1 TO 5
30 READ a(i),b(i)
40 NEXT
50 FOR i=1 TO 5
60 READ p$(i)
70 NEXT
80 DATA 1,7,3,9,2,2,4,6,9,1
90 DATA fred,francois,dany,olivier,eric
100 FOR i=1 TO 5
110 PRINT p$(i),":";a(i)*b(i)
120 NEXT
```

Si la boucle **FOR** de la ligne **20** se change en :

```
20 FOR I=1 TO 4
```

...les deux premiers essais de lecture des chaînes à la ligne **60** donnent alors " **9** " puis " **1** ". Bien qu'elles soient valides, ces valeurs ne représentent pas vraiment le résultat attendu ! Pour obtenir les valeurs souhaitées, il faudrait introduire dans le programme les commandes :

```
15 RESTORE 80
45 RESTORE 90
```

La commande **RESTORE** place le pointeur-lecteur de **DATA** à la ligne spécifiée et peut ainsi servir dans une commande conditionnelle pour lire certains blocs de données selon des critères donnés. De cette manière, dans un jeu à plusieurs niveaux possédant plusieurs écrans, les données de chaque écran pourront être extraites en fonction d'une variable (leur niveau, par exemple). Voici un programme de ce type :

```
1000 REM section de dessin sur ecran
1010 IF niveau=1 THEN RESTORE 2010
1020 IF niveau=2 THEN RESTORE 2510
1030 IF niveau=3 THEN RESTORE 3010
1040 FOR y=1 TO 25
1050 FOR x=1 TO 40
1060 READ char
1070 LOCATE x,y:PRINT CHR$(char);
1080 NEXT x,y
:
2000 REM DATA pour l'ecran 1
2010 DATA 200,190,244,244,210 ...etc
:
2500 REM DATA pour l'ecran 2
2510 DATA 100,103,245,243,251 ...etc
:
3000 REM DATA pour l'ecran 3
3010 DATA 190,191,192,193,194 ...etc
```

Les commandes **DATA**, **READ** et **RESTORE** peuvent également être illustrées par un programme musical. Les périodes sonore sont lues (**READ**) dans les commandes **DATA** tandis que **RESTORE** relance une section en ramenant le pointeur au début d'une partie donnée du programme :

```
10 FOR i=1 TO 3
20 RESTORE 100
30 READ note
40 WHILE note <>-1
50 SOUND 1,note,35
60 READ note
70 WEND
80 NEXT
90 SOUND 1,142,100
100 DATA 95,95,142,127,119,106
110 DATA 95,95,119,95,95,119,95
120 DATA 95,142,119,142,179,119
130 DATA 142,142,106,119,127,-1
run
```

Des sons à votre portée...

Parmi les fonctions du 6128, les commandes de son et d'enveloppe vous paraîtront probablement les plus obscures à première vue. Il n'y a absolument pas de quoi. Avec un peu d'entraînement, vous serez capable d'obtenir toute une gamme d'effets sonores différents et même de composer des harmonies.

Commençons par les quatre premières parties de la commande **SOUND** : numéro de canal, période sonore, durée de la note et volume. Vous vous demandez peut-être quelles valeurs chaque nombre peut prendre ?

Pour le moment, nous laisserons la première partie (numéro de canal) de côté, car ce n'est pas la plus simple. La période sonore peut prendre les valeurs 0 à 4095, mais seules quelques-unes de ces valeurs correspondent à des notes de musique (elles sont listées dans la partir 5 du chapitre « Pour information... »). Ainsi, le nombre 239 correspond au Do médium et 253 à la note située juste en-dessous : le Si. Mais le son obtenu avec les valeurs 240 à 252 ne correspond à aucune des notes du piano. Aucune note n'est jouée lorsque la période est fixée à 0, ce qui nous sera bien utile lorsque nous passerons au « bruit » (voir plus loin).

La troisième partie de la commande **SOUND** fixe la durée de la note en centièmes de seconde. Les valeurs admissibles se situent entre 1 et 32767 (inclus). Toutefois, pour une valeur nulle (zéro), la durée de la note est déterminée par l'enveloppe sélectionnée (on y reviendra). Une valeur négative indique qu'il faut répéter l'enveloppe le nombre de fois spécifié. Ainsi, -3 signifie « répète l'enveloppe de volume 3 fois » (nous y reviendrons également plus tard).

La quatrième partie de la commande est le volume. Il prend les valeurs 0 à 15 (12 par défaut). Dans les sons que nous avons entendus jusqu'à présent, le volume était constant pendant toute l'exécution de la note. Lorsque la note est modulée par une « enveloppe de volume », la valeur indiquée dans la commande **SOUND** concerne le volume au départ de la note.

Voyons maintenant le numéro de canal. Autant vous dire dès à présent qu'il correspond à un nombre binaire - il va donc vous falloir étudier un peu les nombres binaires afin d'en comprendre la signification (voir la première partie de ce chapitre).

Vous avez le choix entre trois canaux pour jouer un son. Si votre ordinateur est connecté à un amplificateur stéréophonique, l'un des canaux va sur la sortie gauche, l'autre sur la sortie droite et le troisième sur les deux à la fois (au milieu). Les numéros suivants sont utilisés pour indiquer le canal sur lequel une note doit être jouée :

1 canal A
2 canal B
4 canal C

Lorsque vous désirez jouer sur plusieurs canaux, il vous suffit d'additionner les numéros des canaux désirés. Ainsi, pour jouer sur A et C, vous indiquerez 1 + 4 = 5.

```
SOUND 5,284
```

Vous vous demandez certainement pourquoi on a attribué au canal C le numéro 4 au lieu de 3. Cela tient au fait que l'on n'a utilisé ici que des puissances de 2 (1 = 2↑0, 2 = 2↑1, 4 = 2↑2), si bien que leurs combinaisons ont la structure d'un nombre binaire. Dans un nombre binaire de trois chiffres, chacun des chiffres peut prendre les valeurs 0 ou 1, déterminant ainsi l'état en service ou hors service du canal correspondant. Ainsi, dans notre exemple :

5 en notation décimale équivaut à **1*4 + Ø*2 + 1*1**, donc 101, en notation binaire. En associant un chiffre binaire à chaque colonne A, B ou C on obtient :

C B A
1 0 1

En d'autres termes, le canal C est en service, le canal B est hors service et le canal A est en service. Si l'on désire jouer une note sur les canaux A et B il faut indiquer :

C B A
0 1 1

Le nombre binaire 011 correspondant à **Ø*4 + 1*2 + 1*1 = 3**, la commande **SOUND** prend la forme :

```
SOUND 3,142
```

Naturellement, on obtient exactement le même résultat en additionnant simplement les numéros de canal (souvenez-vous que A = 1, B = 2 et C = 4). Le numéro à indiquer pour jouer sur les canaux A et B est donc bien 1 + 2 = 3.

Ne vous inquiétez surtout pas si vous n'avez pas compris cette démonstration. La seule chose vraiment importante à retenir pour choisir une combinaison de canaux est l'addition des numéros de canal.

Hélas, 1, 2 et 4 ne sont pas les seuls paramètres de numéro de canal. Les valeurs 8, 16 et 32 peuvent également être utilisées, afin d'indiquer qu'un son produit sur un canal a rendez-vous avec un autre canal (A, B et C respectivement). Qu'entendons-nous par rendez-vous ? C'est très simple : les sons que nous avons fabriqués jusqu'à présent étaient immédiatement joués sur le canal indiqué. Essayez donc ce qui suit :

```
SOUND 1,142,2000
SOUND 1,90,200
```

A moins d'être particulièrement lent au clavier, vous avez constaté que vous aviez le temps de taper la seconde commande avant que la première n'ait fini de s'exécuter. Ceci est dû au fait que le mode sonore est capable de maintenir jusqu'à 5 commandes SOUND en file d'attente pour un même canal. Supposons que nous désirions jouer un son sur le canal A, puis deux sons simultanés sur les canaux A et B. Il faudrait bien trouver un moyen d'indiquer à l'ordinateur qu'il ne doit pas émettre le son du canal B avant que le canal A ne soit prêt à entamer sa seconde note. C'est alors qu'intervient la notion de rendez-vous, que l'on obtient de deux façons différentes :

```
SOUND 1,200,1000
SOUND 3,90,200
```

La seconde note est ici dirigée sur A et B, elle ne peut donc être jouée qu'après la fin de la note du canal A. On fait ainsi attendre une note jusqu'à ce que les canaux qu'elle doit utiliser soient libres. Inconvénient évident : le même son sera dirigé sur chacun des canaux (le son **,90,200**, dans notre cas, est joué par A et B). Voici l'autre méthode :

```
SOUND 1,200,2000
SOUND 1+16,90,200
SOUND 2+8,140,400
```

Dans cet exemple, on fixe à la seconde note de A un rendez-vous avec la note de B (et à la note de B un rendez-vous avec celle de A). L'avantage est clair : les deux notes pourraient être différentes, elles seraient néanmoins liées de sorte qu'aucune ne pourrait être émise sans que les deux canaux ne soient libres. Voilà donc ce qu'on appelle un rendez-vous. Là encore, la signification de ces valeurs n'apparaît qu'en binaire :

8=2↑3, 16=2↑4, et 32=2↑5.

Nous pouvons donc maintenant nous représenter le numéro de canal sous la forme d'un nombre binaire dont les chiffres portent les titres suivants :

Rendez-vous avec C	Rendez-vous avec B	Rendez-vous avec A	Jouer sur C	Jouer sur B	Jouer sur A
Ajouter 32	Ajouter 16	Ajouter 8	Ajouter 4	Ajouter 2	Ajouter 1

Ainsi, pour jouer sur C une note ayant rendez-vous avec A, vous spécifiez :

0	0	1	1	0	0

C'est-à-dire le nombre 1100, correspondant en décimale à 8+4=12.

Le numéro de canal 12 indiquera donc à l'ordinateur qu'avant de jouer une note sur le canal C il doit attendre la note du canal A avec laquelle il a rendez-vous.

Si l'on ajoute au numéro de canal le nombre 64 (2↑6), l'ordinateur retient la note sans la jouer. Cette valeur indique en effet que la note n'est libérée que lors d'une commande **RELEASE**.

Enfin, si l'on ajoute 128 (2↑7), la file d'attente du canal indiqué est annulée. Vous pouvez utiliser cette valeur pour arrêter rapidement, sur un canal donné, un son qui risque de durer trop longtemps :

```
SOUND 1,248,30000       (ce son durerait cinq minutes, mais)
SOUND 1+128,0           (lui coupe le sifflet)
```

En mode direct, le moyen le plus rapide d'arrêter un son est d'appuyer sur la touche **[DEL]** en début de ligne : le signal sonore d'avertissement vide alors immédiatement tous les canaux sonores.

Maintenant que vous savez diriger un son sur l'un des trois canaux (en fixant au besoin des rendez-vous), vous aimeriez certainement apprendre à produire des sons un peu plus mélodieux que ceux auxquels vous limite la commande **SOUND** telle que vous l'employez. Il vous faut pour cela attribuer au son une enveloppe, c'est-à-dire un modèle définissant la variation du volume en cours d'exécution. Lorsqu'une note est jouée sur un instrument, on a d'abord une attaque pendant laquelle le volume augmente très rapidement, après quoi il descend à un niveau inférieur, auquel il se maintient un certain temps avant de s'éteindre progressivement. Il est parfaitement possible de reproduire cette dynamique sonore au moyen de la commande **SOUND**. On utilise pour cela la commande associée **ENV**. Commençons par un exemple très simple :

```
ENV 1,5,3,4,5,-3,8
SOUND 1,142,0,0,1
```

La commande **ENV** est spécifiée avant la commande **SOUND** à laquelle elle s'applique. Pour utiliser une enveloppe avec la commande SOUND, on indiquera son numéro au cinquième paramètre de cette dernière (dans l'exemple ci-dessus, nous avons choisi l'enveloppe numéro 1). Le premier paramètre d'une commande **ENV** est son numéro. Du fait que la commande **ENV** indique la durée et l'intensité d'une note, on fixera à 0 les paramètres de la commande **SOUND** correspondants. L'enveloppe ci-dessus définit une augmentation de volume en cinq pas correspondant à une augmentation de 3 unités et durant 4 centièmes de seconde chacune. Le son diminue ensuite en 5 pas de 3 unités et de 8 centièmes de seconde. Ainsi, le premier nombre, ou « numéro d'enveloppe », est suivi de deux groupes de trois nombres. A l'intérieur d'un groupe se trouve un premier nombre indiquant en combien de pas s'effectuera l'augmentation ou la diminution de volume, un deuxième indiquant la variation de volume au cours de chaque pas, et le troisième la durée de ces pas.

La durée totale correspond au premier nombre (le nombre de pas) multiplié par le troisième (la durée d'un pas). La variation totale d'amplitude du volume est égale à celle de chaque pas multipliée par le nombre de pas. La durée totale d'une enveloppe comprenant plusieurs segments est égale à la somme des durées de chacun des segments.

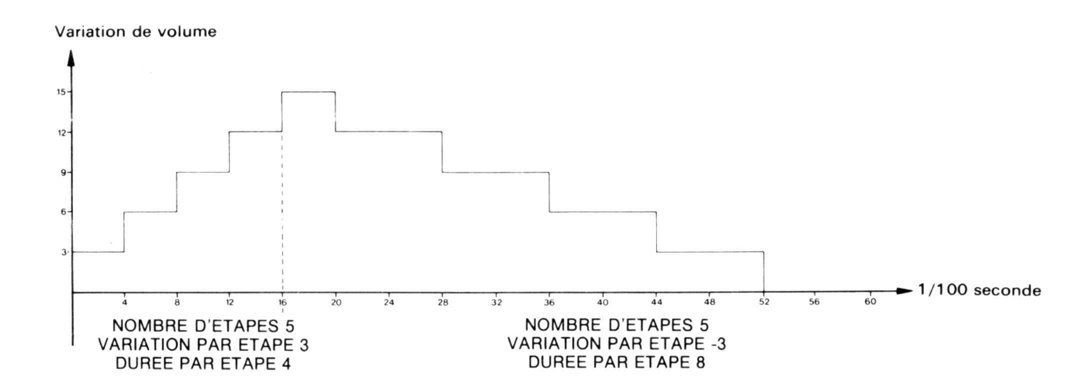
Le volume initial (défini dans la commande **SOUND**), peut bien sûr être différent de **Ø**. L'enveloppe définie dans le premier exemple formait un crescendo suivi d'un decrescendo. Voici maintenant un decrescendo suivi d'un crescendo :

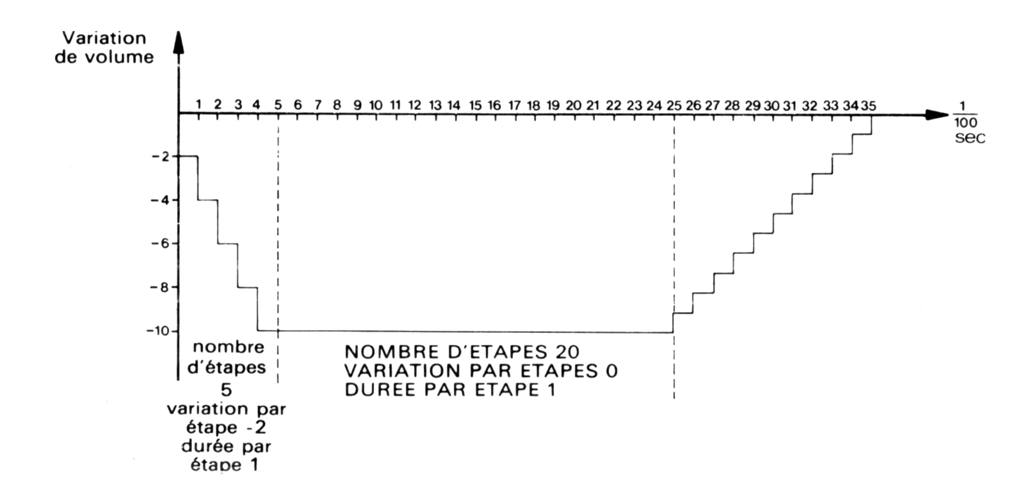
```
ENV 2,5,-2,1,20,0,1,10,1,1
SOUND 1,248,0,15,2
```

Cette enveloppe, qui portera le numéro **2**, est composée de trois segments. Dans le premier, le volume décroît en **5** pas de **-2**. Il diminue donc de 2 à chaque pas d'un centième de seconde. Le second segment comprend **2Ø** pas, au cours desquels le taux de variation est de **Ø** (le volume est donc constant). Là encore, la durée d'un pas est de 1 centième de seconde. Le troisième segment, enfin, comprend **1Ø** segments augmentant le volume de **1** chacun et durant également 1 centième de seconde.

Comme la commande **SOUND** indique un volume initial de **15**, ce dernier sera donc de **5** après le premier segment, il restera à ce niveau pendant 20 centièmes de seconde, avant de remonter jusqu'à **15** au cours du dernier segment de l'enveloppe.

Si cette représentation chiffrée de l'enveloppe vous semble trop abstraite, vous préférerez peut-être la dessiner d'abord sur un graphe, puis en extraire les valeurs à introduire dans la commande **ENV**. Voici la représentation graphique des deux enveloppes définies jusqu'à présent :

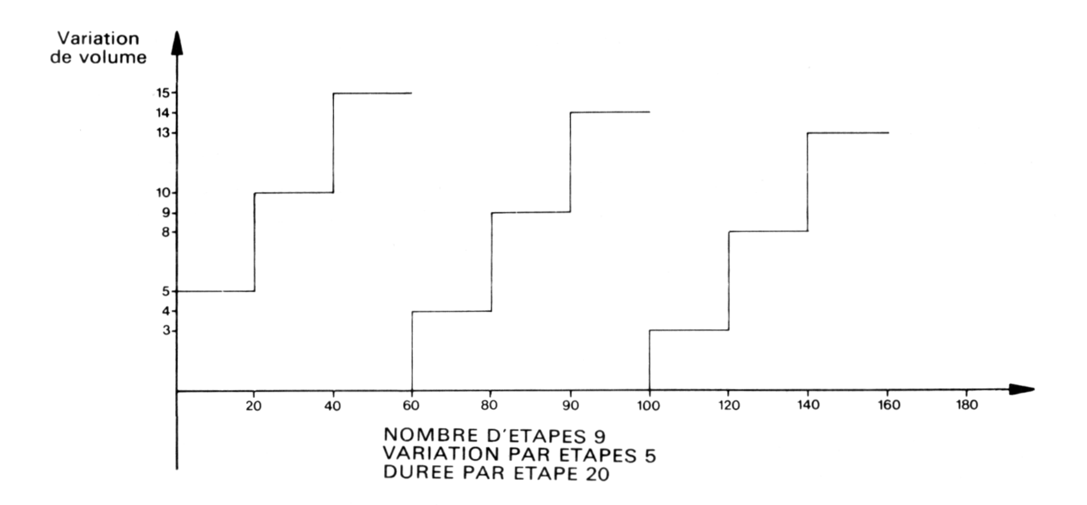




Une enveloppe pouvant contenir jusqu'à cinq segments de trois valeurs chacun, l'instruction **ENV** peut donc utiliser un maximum de 16 paramètres (en comptant le numéro d'identification de l'enveloppe, compris entre 1 et 15). Lorsque les variations de volume indiquées excèdent les limites d'intensité autorisées (entre 0 et 15), la valeur de l'intensité boucle sur elle-même, si bien que la première valeur dépassant 15 correspond au volume 0 et la première au-dessous de 0, au volume 15 :

```
ENV 3,9,5,20
SOUND 1,142,0,0,3
```

Cette enveloppe très simple définit **9** pas au cours desquels le volume augmente de **5** en **5** pour une durée de **20** centièmes de secondes chacun. Le volume atteint le niveau 15 à la fin du troisième pas, si bien que la variation suivante amène celui-ci à 4, la suivante à 9 etc... Voici, sous forme graphique, l'effet produit par cette commande :



Le nombre des pas peut varier de 0 à 127 et leur amplitude de -128 à +127 (une valeur négative correspondant à une diminution de volume). La durée des pas peut prendre des valeurs allant de 0 à 255.

Nous savons maintenant modeler à volonté l'enveloppe de volume. Voyons à présent comment jouer sur la hauteur de la note pour créer, par exemple, un effet de vibrato, c'est-à-dire une note oscillant autour d'une hauteur donnée.

La procédure à suivre ressemble beaucoup à celle de l'enveloppe de volume. On utilise la fonction **ENT**, dont voici un exemple.

```
ENT 1,5,1,1,5,-1,1
SOUND 1,142,10,15,,1
```

Le numéro d'enveloppe de tonalité constitue le sixième paramètre de la commande **SOUND**. Là encore, la commande **ENT** doit intervenir avant la commande **SOUND**.

La commande **ENT** de ce premier exemple précise que l'enveloppe numéro **1** est composée de **5** pas augmentant la période d'une unité et durant un centième de seconde chacun. La partie suivante définit également **5** pas diminuant de **1** la période et durant un centième de seconde chacun. La durée totale de l'enveloppe est de 5 + 5 = 10. Vous remarquez que nous avons dû indiquer cette durée à l'intérieur de la commande **SOUND**. En effet, à la différence de **ENV**, la commande **ENT** ne précise pas la durée de la note jouée. Si la durée indiquée dans la commande **SOUND** est plus courte que celle de l'enveloppe de ton, une partie de cette dernière sera perdue. Si, au contraire, elle est plus longue, la fin de la note ne subira aucune variation de période. Ceci est également valable lorsque l'enveloppe de volume détermine la durée de la note.

(Remarquez également l'absence du paramètre « enveloppe de volume », cette enveloppe n'étant pas créée pour notre exemple).

La durée de l'enveloppe de tonalité est, dans la plupart des cas, beaucoup plus courte que celle de la note. On indique alors qu'elle doit se reproduire jusqu'à la fin de la note, en choisissant un numéro d'enveloppe négatif. Le paramètre utilisé dans la commande SOUND est le nombre positif correspondant :

```
ENT -5,4,1,1,4,-1,1
SOUND 1,142,100,12,,5
```

On obtient alors un effet de vibrato. Il est souvent préférable de définir une enveloppe de tonalité assurant une variation symétrique autour de la fréquence initiale, évitant ainsi que la note ne s'éloigne de plus en plus de celle-ci. Essayez par exemple :

```
ENT -6,3,1,1
SOUND 1,142,90,12,,6
```

La très nette chute de fréquence que vous avez constatée provient du fait que chaque répétition de l'enveloppe augmente la période de la note de **3** unités, ce qui se produit 30 fois (90 divisé par trois). Cette caractéristique permet des effets de sirène ou de pédale « wah-wah » :

```
ENT -7,20,1,1,20,-1,1
SOUND 1,100,400,12,,7

ENT -8,60,-1,1,60,1,1
SOUND 1,100,480,12,,8
```

On peut définir ainsi jusqu'à quinze enveloppes numérotées de 1 à 15, en utilisant les valeurs négatives correspondantes pour obtenir la répétition de l'enveloppe. Le nombre de pas, premier paramètre de chaque groupe, peut varier entre 0 et 239. L'amplitude du pas prend, comme pour la commande **ENV**, des valeurs comprises entre -128 et 127, tandis que la durée par pas doit être comprise entre 0 et 255. On est là encore limité à cinq segments composés de trois valeurs chacun.

Il est possible d'ajouter un dernier paramètre à la commande **SOUND** : le niveau de bruit que l'on désire incorporer au son. Lorsqu'on utilise un bruit, il ne faut jamais oublier que cette fonction ne dispose que d'un canal et que, de ce fait, chaque nouvelle période recouvrera la précédente.

On peut utiliser un bruit soit en le mélangeant à un son pour modifier celui-ci, soit isolément en donnant à la période sonore (deuxième paramètre de la commande **SOUND**) la valeur **0**. C'est ainsi que l'on reproduit les effets de percussions. Essayez par exemple :

```
ENT -3,2,1,1,2,-1,1
ENV 9,15,1,1,15,-1,1
FOR a=1 TO 10: SOUND 1,4000,0,0,9,3,15:NEXT
```

Vous constatez que ces instructions, éléments de départ pour la simulation d'un bruit de train, combinent le bruit avec les deux types d'enveloppes.

Les paramètres de durée et de volume étant fixés à 0 dans la commande **SOUND**, ces deux caractéristiques sont donc déterminées par l'enveloppe de volume.

Si l'utilisation des commandes **SOUND**, **ENV** et **ENT** est parfaitement claire pour vous, nous allons pouvoir aborder les différentes commandes et fonctions qui leur sont associées.

Lorsque nous parlions du numéro de canal de la commande **SOUND**, nous avions dit qu'en lui ajoutant le nombre 64 on retenait le son dans une file d'attente, jusqu'à ce qu'il soit libéré par la commande **RELEASE**. Le mot clé **RELEASE** est suivi d'un nombre correspondant au bit significatif de celui des trois canaux que vous désirez libérer. La seule chose importante à retenir est que :

4 désigne le canal C
2 désigne le canal B
1 désigne le canal A

...et que vous obtiendrez la libération de plusieurs canaux à la fois en additionnant les nombres correspondants. Ainsi, pour libérer les sons retenus sur les trois canaux, on indiquera :

```
RELEASE 7
```

...le nombre **7** étant égal à 1 + 2 + 4. L'ordinateur ne tient pas compte d'une commande **RELEASE** si aucun des canaux ne se trouve bloqué. Essayez maintenant :

```
SOUND 1+64,90
SOUND 2+64,140
SOUND 4+64,215
RELEASE 3: FOR t=1 TO 1000:NEXT:RELEASE 4
```

Les sons définis par les commandes **SOUND** ne sont produits qu'au moment où la première commande **RELEASE** libère les canaux A et B. Le son retenu dans le canal C est libéré à la fin de la boucle d'attente.

Il existe encore un autre moyen d'obtenir un rendez-vous de plusieurs sons. Lorsqu'un son est placé en file d'attente dans un canal à l'état bloqué (numéro de canal + 64), ce dernier retiendra également tous les sons dirigés ensuite sur cette file. Si plus de quatre sons supplémentaires sont ajoutés à cette même file, la machine effectue une pause sous la forme, par exemple, d'un sous-programme appelé après un laps de temps déterminé par la commande **AFTER** ou **EVERY**. Cette méthode est donc à déconseiller en mode sonore, puisqu'elle provoque une pause du programme dès qu'une file se trouve surchargée. On rencontre le même inconvénient lorsque de nombreux sons de longue durée sont accumulés à un rythme trop rapide. Voyez plutôt :

```
10 FOR a=1 TO 8
20 SOUND 1,100*a,200
30 NEXT
40 PRINT"bonjour"
run
```

Vous constatez que le mot " **bonjour** " ne s'affiche qu'au bout d'un certain temps, après exécution des trois premiers sons. Ce retard tient au fait que l'exécution du programme ne peut continuer tant qu'une place ne s'est pas libérée dans la file.

Le BASIC possède un mécanisme d'interruption assez similaire à celui qu'utilisent les commandes **AFTER**, **EVERY** ou **ON BREAK GOSUB**. Il donne accès à un sous-programme d'émission sonore qui n'est appelé que lorsqu'une place se libère dans la file demandée. En voici un exemple :

```
10 a=0
20 ON SQ(1) GOSUB 1000
30 PRINT a;
40 GOTO 30
1000 a=a+10
1010 SOUND 1,a,200
1020 IF a<200 THEN ON SQ(1) GOSUB 1000
1030 RETURN
run
```

Ce programme, comme vous pouvez le constater, n'effectue jamais de pause. La commande **SOUND** n'est appelée que si la file d'attente du canal A (numéro 1) contient une place libre, ce que la commande **ON SQ(1) GOSUB** est chargée de vérifier à la ligne **20**. Cette commande initialise un mécanisme d'interruption qui appelle le sous-programme sonore lorsqu'un espace libre apparaît dans la file. La commande **ON SQ GOSUB** doit être réinitialisée après appel du sous-programme, ce que fait la ligne **1020**. Dans notre exemple, le sous-programme sonore se réinitialise tant que « a » est inférieur à 200.

Il est possible de demander à l'ordinateur de jouer une musique de fond tout en exécutant un programmme complet (déplacement d'objets à l'écran, calculs etc...). Il suffit pour cela de faire jouer les notes par un sous-programme qui sera appelé seulement lorsque la file d'attente comportera une place libre. Le programme n'effectuera jamais de pause en attendant qu'une place se libère. En fournissant les notes à ce sous-programme sous forme d'instructions **DATA**, il serait très simple de lui faire cesser de se réinitialiser juste avant de lire la dernière donnée.

Le nombre placé entre les parenthèses de la commande **ON SQ( ) GOSUB** est 1, 2 ou 4 selon le canal dont la file d'attente doit être sollicitée.

Il existe une fonction **SQ( )** permettant d'interroger l'état d'un canal sonore donné, qu'on désignera de même par le nombre 1,2 ou 4. Cette fonction restitue un nombre dont chaque bit porte une signification et dont l'interprétation nécessite quelques connaissances des nombres binaires. Les bits contenus dans les valeurs obtenues ont la signification suivante :

BIT	DECIMAL	SIGNIFICATION
0,1,2	1,2,4	Nombre de places libres dans la file
3	8	Rendez-vous avec A pour la note en tête de file
4	16	Rendez-vous avec B pour la note en tête de file
5	32	Rendez-vous avec C pour la note en tête de file
6	64	File bloquée
7	128	Une note en cours d'exécution

Voyons un exemple très simple :

```
10 SOUND 2,200
20 x=SQ(2)
30 PRINT BIN$(x)
run
```

Le nombre binaire obtenu (**10000100**) contient le bit 7 indiquant que le canal était en activité au moment où la fonction **SQ** a été utilisée. Les trois derniers chiffres (100) correspondent au nombre décimal 4 : il reste donc quatre places libres dans la file. Cette fonction permet de connaître l'état d'une file à un point donné du programme, alors que **ON SQ( ) GOSUB** interroge celui-ci, puis réagit en conséquence à un moment imprévisible.

Les exemples cités jusqu'à présent ne prévoyaient l'émission que d'une, voire deux notes à la fois. Nous allons maintenant considérer la possibilité de manier des ensembles de notes indépendantes les unes des autres (un morceau de musique, par exemple) en les entrant sous forme d'une commande **DATA** lue (**READ**) par la commande **SOUND**.

```
10 FOR octave=-1 TO 2
20 FOR x=1 TO 7:REM notes par octave
30 READ note
40 SOUND 1,note/2↑octave
50 NEXT
60 RESTORE
70 NEXT
80 DATA 426,379,358,319,284,253,239
run
```

Notre dernier exemple de programme sonore montre ce qu'il est possible de réaliser à partir de ces éléments. Un rythme et une mélodie sont joués sur les canaux A et B synchronisés au moyen de rendez-vous. L'exemple montre comment introduire dans une intruction **DATA** des informations concernant les notes, les octaves, les durées et les rendez-vous :

```
10 REM ligne 190 donne la clef de SOL
20 REM ligne 200 donne la clef de FA
30 DIM gamme%(12):FOR x%=1 TO 12:READ gamme%(x%):NEXT
40 ca1%=1:READ ca1$:ca2%=1:READ ca2$
50 CLS
60 vit%=12
70 gamme$=" a-b b c+c d-e e f+f g+g"
80 ENV 1,2,5,2,8,-1,10,10,0,15
90 ENV 2,2,7,2,12,-1,10,10,0,15
100 ENT -1,1,1,1,2,-1,1,1,1,1
110 DEF FNm$(s$,s)=MID$(s$,s,1)
120 ca1%=1:GOSUB 200
130 ca2%=1:GOSUB 380
140 IF ca1%+ca2%>0 THEN 140
150 END
160 DATA &777,&70c,&6a7,&647,&5ed,&598
170 DATA &547,&4fc,&4b4,&470,&431,&3f4
180 DATA 4cr4f4f1f1g1a1-b2c2f4g2g1a1-b6a2cr1f1g1a1-b1a1-b2c2
g2a2g2f1g1a2g2f6e2c2e2c2g2e2c1-b1a2g2f4e4d8c4f3f1c2d4-b2fr2-
b2a2g2f6e2gr
4c4-b1a1f1-b1g2c2-b4a4g4fr6a2a2-b4b2ar2-b2a2g2f6e2g4c4-b1a1f
1-b1g2c2-b4a4g8f.
190 DATA r4f4f8f4e4c4fr8f4e2f2e4d2e2d8c8c6e2f4g4g8e4f3f1c4dr
8g4cr4e4c6f2d4c4c8fr8-e4dr8g8c4e4c6f2d4c4c8f.
200 REM envoi le son sur le canal A
210 p1$=FNm$(ca1$,ca1%)
220 IF p1$<>"r" THEN r1%=0:GOTO 240
230 r1%=16:ca1%=ca1%+1:p1$=FNm$(ca1$,ca1%)
```

```
240 IF p1$="." THEN ca1%=0:RETURN ELSE l1%=VAL(p1$)
250 ca1%=ca1%+1
260 n1$=FNm$(ca1$,ca1%)
270 ca1%=ca1%+1
280 IF n1$="+" OR n1$="-" THEN 350
290 n1$=" "+n1$
300 nd1%=(1+INSTR(gamme$,LOWER$(n1$)))/2
310 IF ASC(RIGHT$(n1$,1))>96 THEN o1%=8 ELSE o1%=16
320 SOUND 1+r1%,gamme%(nd1%)/o1%,vit%*l1%,1,1
330 ON SQ(1) GOSUB 200
340 RETURN
350 n1$=n1$+FNm$(ca1$,ca1%)
360 ca1%=ca1%+1
370 GOTO 300
380 REM envoi le son sur le canal B
390 p2$=FNm$(ca2$,ca2%)
400 IF p2$<>"r" THEN r2%=0:GOTO 420
410 r2%=8:ca2%=ca2%+1:p2$=FNm$(ca2$,ca2%)
420 IF p2$="." THEN ca%=0:RETURN ELSE l2%=VAL(p2$)
430 ca2%=ca2%+1
440 n2$=FNm$(ca2$,ca2%)
450 ca2%=ca2%+1
460 IF n2$="+" OR n2$="-" THEN 530
470 n2$=" "+n2$
480 nd2%=(1+INSTR(gamme$,LOWER$(n2$)))/2
490 IF ASC(RIGHT$(n2$,1))>96 THEN o2%=4 ELSE o2%=8
500 SOUND 2+r2%,gamme%(nd2%)/o2%,vit%*l2%,0,2
510 ON SQ(2) GOSUB 380
520 RETURN
530 n2$=n2$+FNm$(ca2$,ca2%)
540 ca2%=ca2%+1
550 GOTO 480
run
```

Si nous parlions graphiques ?

La partie qui suit est consacrée aux fonctions graphiques de Schneider 664, dont nous allons, à partir d'un premier exemple, découvrir tour à tour les points essentiels.

Nous allons commencer par partager l'écran en deux : une fenêtre de texte (ligne **40**) et une fenêtre graphique (ligne **30**). Puis nous choisirons un **MODE**, ainsi que deux couleurs clignotantes (ligne **20**).

```
10 REM MASK et TAG dans les fenetres
20 MODE 1:INK 2,10,4:INK 3,4,10
30 ORIGIN 440,100,440,640,100,300
40 WINDOW 1,26,1,25
50 CLG 2
```

En exécutant ce programme, vous allez voir apparaître un carré clignotant au milieu de la moitié droite de l'écran. La ligne 50 attribue à ce carré l'encre numéro deux (bleu sombre et rose alternés). La ligne 30 positionne le point d'origine des coordonnées sur le coin inférieur gauche du carré tandis que la commande **MODE** y place le curseur (X=0, Y=0). On peut maintenant tracer une diagonale à l'aide de la ligne **6Ø** :

```
60 DRAW 200,200,3
```

Exécutez le programme, puis ajoutez la ligne suivante :

```
80 MOVE 0,2:FILL 3
```

La ligne **8Ø** pose le curseur graphique à l'intérieur d'une des moitiés du carré, qu'elle colore au moyen de l'encre **3**. Les limites de ce coloriage sont le bord de la fenêtre (qui est aussi celui de notre carré) et tout élément graphique tracé par le stylo graphique utilisé (le **3**) ou pour lequel on a indiqué la même encre que celle du remplissage (la numéro **3** également).

Exécutez le nouveau programme :

Vérifiez ce qui a été dit à propos des limites du coloriage, en écrivant la ligne **7Ø**.

```
70 GRAPHICS PEN 1
run
```

Remarquez bien que si l'encre de la diagonale n'était pas la même que celle du coloriage, c'est le carré tout entier qui serait colorié, ce que vous pouvez vérifier en remplaçant **FILL 3** par **FILL 1** dans la ligne **8Ø**. Revenez ensuite à l'état initial (**FILL 3**).

Dessinons maintenant un cadre :

```
100 MOVE 20,20
110 DRAW 180,20
120 DRAW 180,180
130 DRAW 20,180
140 DRAW 20,20
run
```

L'encre 1 du cadre a été indiquée par la ligne **70**. Si celle-ci n'existait pas, il nous faudrait ajouter l'indication " **,1** ", soit comme troisième paramètre de la commande **MOVE** en ligne **100**, soit comme troisième paramètre de la commande **DRAW** en ligne **110**. L'ordinateur saurait alors qu'il lui faut changer de stylo graphique.

Du point à la ligne...

Une ligne peut être pleine, ou en pointillés. La commande **MASK** permet de déterminer la taille des pointillés. On définit pour cela une séquence se reproduisant tous les huit pixels et se poursuivant ainsi de ligne en ligne. Il est possible de revenir au début de cette séquence par l'intermédiaire d'une nouvelle commande **MASK** reprenant les mêmes paramètres.

La séquence est représentée au moyen d'un octet dont la disposition interne détermine l'emplacement de l'encre. Nous utiliserons dans notre exemple un nombre binaire constant (indiqué par le code &X) pour définir un segment dont seuls les quatre pixels centraux seront dessinés, les deux pixels de chaque extémité n'apparaissant pas. Cette disposition nous donnera une suite de tirets de quatre pixels espacés de quatre pixels. Voici la ligne qui accomplit ce prodige :

```
90 MASK &X111100
run
```

Mais pourquoi ce programme ne respecte-t'il pas la séquence choisie lorsqu'il trace un coin ? Tout simplement parce que le coin est tracé deux fois : en tant que dernier point et premier point des lignes qui le composent. On peut contourner ce problème comme ceci :

```
115 MOVE 180,22
125 MOVE 178,180
135 MOVE 20,178
run
```

On obtient bien l'effet désiré, mais il y a beaucoup plus simple. Si l'on ajoute à la commande **MASK** un second paramètre " **,0** ", l'ordinateur ne trace pas le premier point de chaque ligne. Corrigeons alors la ligne **90** :

```
90 MASK &00111100,0
```

...et effaçons les lignes que nous venions d'ajouter en entrant :

```
115
125
135
```

Le cadre tracé en pointillés par notre programme a maintenant un aspect symétrique. Si le second paramètre de **MASK** était fixé à " **,1** ", la commande tracerait à nouveau le premier pixel de chaque ligne.

Mais regardez de plus près les espaces séparant les tirets : ceux du triangle inférieur ont quelque chose de plus que ceux de triangle supérieur. Il s'agit du papier graphique auquel la commande **CLG 2**, ligne **50**, a attribué l'encre **2**. L'encre **2** étant aussi la couleur du fond, le papier graphique est invisible dans le triangle supérieur. Modifions donc la ligne **50** :

```
50 CLG 2:GRAPHICS PAPER 0
```

...et faisons tourner le programme : la couleur du papier se détache maintenant tout autour du cadre.

On peut également choisir un papier graphique « transparent » si l'on désire que les espaces d'une ligne pointillée laissent transparaître le tracé sous-jacent. On obtient un tracé transparent en ajoutant à la commande **GRAPHICS PEN** le paramètre " **,1** " (le paramètre " **,0** " nous ramène au tracé opaque). Modifiez donc la ligne **70** :

```
70 GRAPHICS PAPER 1,1
run
```

...et voyez le résultat.

Sur la position du curseur graphique, il est non seulement possible de tracer des lignes ou des points, mais aussi d'inscrire du texte. Cette caractéristique permet de positionner les caractères avec une bien plus grande précision (à un pixel près, au lieu de huit) et de les agrémenter en jouant sur les différents modes d'encre graphique (voir plus haut).

Pour écrire sur l'écran graphique, positionnez d'abord le curseur à l'endroit où devra se trouver le caractère de gauche, puis entrez la commande **TAG** (ou **TAG#1, 2** etc... selon le canal désiré) suivie des commandes **PRINT** usuelles. Le curseur graphique se déplace automatiquement de 8 pixels vers la droite à chaque caractère tracé. Essayez vous-même :

```
160 MOVE 64,108
170 TAG
180 PRINT"SALLY"
190 TAGOFF
run
```

Bien que les messages émis par le BASIC s'affichent sur la fenêtre de texte sans tenir compte de l'indicateur **TAG/TAGOFF**, il est bon de s'habituer à annuler la commande **TAG** dès que l'on n'a plus besoin d'elle.

Mais que représentent ces flèches à droite du nom ? Ce sont les symboles du retour chariot **CHR$(13)** et du changement de ligne **CHR$(10)**. Le logiciel graphique convertit les 32 premiers caractères ASCII sous leur forme visualisable, comme s'ils étaient dirigés sur la fenêtre texte précédés de **CHR$(1)**. La raison en est que ces 32 caractères de contrôle ne sont utiles qu'à l'intérieur de la fenêtre texte. Dans le même ordre d'idées, aucun bouclage du texte n'est prévu lorsqu'on dépasse la marge droite de la fenêtre graphique.

Pour éviter l'affichage des symboles de retour chariot et de changement de ligne, on a recours au point-virgule à la fin de la commande **PRINT** :

```
180 PRINT"SALLY";
run
```

Le texte dirigé sur une fenêtre graphique est conditionné par les commandes **GRAPHICS PEN** régissant le tracé de lignes. Nous trouvant sous la commande **GRAPHICS PEN 1**, le nom est donc écrit en mode transparent. Si nous passons à la commande :

```
150 GRAPHICS PEN 1,0
run
```

...nous aurons de nouveau le papier opaque, tandis que :

```
150 GRAPHICS PEN 0,1
run
```

...écrira avec l'encre numéro **0** (transparente).

Exécutez maintenant le programme après avoir effacé la ligne **150**. Vous remarquez alors que le stylo graphique a retrouvé l'encre numéro **1** + le mode transparent indiqués en ligne **70**.

Les caractères transparents

Il est possible, au moyen de certains codes de contrôle, d'écrire sur l'écran de texte en caractères transparents. Ajoutez au programme les lignes :

```
200 PRINT #2,CHR$(22);CHR$(1)
210 LOCATE #2,32,14:PRINT #2,"******"
220 LOCATE #2,32,14:PRINT #2,"______"
230 PRINT #2,CHR$(22);CHR$(0)
run
```

La ligne **200** établit le mode transparent sur le canal #**2**. Vous pouvez constater que le soulignement apparaît « par dessus » les astérisques : il est donc possible de superposer plusieurs caractères, même de couleur différente. La ligne **230** annule le mode transparent et le canal #**2** revient donc au mode opaque.

Les modes d'encre

Il existe un mode d'encre graphique permettant l'interaction entre l'encre utilisée et celle qui se trouve déjà sur l'écran. L'ordinateur calcule alors pour chaque pixel une encre résultante obtenue par combinaison logique des deux encres (stylo ou papier) utilisées. On dispose ici des opérateurs logiques XOR, AND et OR. Le mode d'encre peut être défini soit par le quatrième paramètre des commandes **DRAW/DRAWR**, **PLOT/PLOTR** ou **MOVE/MOVER**, soit en ajoutant à la commande **PRINT** la séquence de contrôle **CHR$(23)** ; **CHR$**(<mode>). Dans tous les cas, **1** correspond à XOR (OU exclusif), **2** à AND (ET) et **3** à OR (OU). La valeur **Ø** rétablit le mode normal, dans lequel la nouvelle encre utilisée apparaît « telle quelle ».

On verra dans l'exemple qui suit une utilisation de l'opérateur XOR. Cet opérateur a la propriété de restituer l'image originale lorsqu'on trace deux fois le même motif et est donc souvent employé avec la tortue graphique. Vous constatez par ailleurs que le sous-programme traçant le carré est exécuté deux fois (lignes **11Ø** et **13Ø**), ainsi que la commande **PRINT** associée à **TAG** (ligne **17Ø** et **19Ø**). La commande **FRAME** provoque un temps d'attente suffisamment long pour nous permettre d'observer l'effet produit. Remarquez que les commandes de la ligne **9Ø** permettent de se dispenser du premier paramètre : celui-ci reste par défaut à sa valeur antérieure.

Le troisième paramètre de la commande MOVE (ligne **22Ø**) donne à la commande **GRAPHICS PEN** la valeur **1**, qui se substitue à la valeur **3** fixée en ligne **6Ø**. Le mode XOR est établi par le quatrième paramètre de la commande **DRAWR** (ligne **23Ø**). Remarquez là encore le paramètre absent.

On peut faire réapparaître l'image recouverte en annulant la commande **MASK** (ligne **9Ø**). Vous voyez également que les coins du carré ont disparu : en effet, après avoir été tracés deux fois (à la fin d'une ligne et au commencement d'une autre), ils ont été annulé par l'opération XOR.

```
10 REM modes d'encre XOR
20 MODE 1:INK 2,10:INK 3,4
30 ORIGIN 440,100,440,640,100,300
40 WINDOW 1,26,1,25
50 CLG 2:GRAPHICS PAPER 0
60 DRAW 200,200,3
70 MOVE 2,0:FILL 3
80 ORIGIN 440,0,440,640,0,400
90 GRAPHICS PEN ,1:MASK ,0
100 FOR y=60 TO 318 STEP 2
110 GOSUB 220
120 FRAME:FRAME
130 GOSUB 220
```

```
140 NEXT
150 TAG
160 FOR y=60 TO 318 STEP 2
170 MOVE 96,y:PRINT CHR$(224);
180 FRAME:FRAME
190 MOVE 96,y:PRINT CHR$(224);
200 NEXT
210 END
220 MOVE 90,y,1
230 DRAWR 20,0,,1
240 DRAWR 0,20
250 DRAWR -20,0
260 DRAWR 0,-20
270 RETURN
run
```

L'animation graphique

Il est possible, en modifiant les couleurs affectées aux encres, d'obtenir un effet d'animation. L'effet de mouvement est ainsi créé alors que le contenu de la mémoire d'écran reste inchangé. Un exemple d'application de ce procédé se trouve d'ailleurs dans le programme de « Bienvenue » de votre disque-système maître (vous pouvez assister à cette démonstration en lançant la commande **RUN " disc "**). Cependant, l'effet de « palette clignotante » utilisé dans cet exemple est insuffisant lorsque les formes successives constituant l'animation doivent se chevaucher. Le programme qui suit inscrit les chiffres 1 à 4 sur l'écran à l'aide d'encres combinées entre elles par l'opérateur OR. Le programme détermine par balayage le graphisme des caractères affichés dans le coin inférieur gauche de l'écran, puis le reproduit sous forme aggrandie. Les nombres successifs sont écrits en combinant les encres 1, 2, 4 et 8 en mode OR. On a eu ici recours à une séquence de caractères de contrôle (ligne **50**).

Les lignes **160** et suivantes font tourner la palette en accord avec une formule mathématique : les nombres reproduits en graphique sont alors affichés un par un. Le choix de l'encre se fait en examinant tour-à-tour chacune d'entre elles pour déterminer si elle contient la séquence binaire donnée : le nombre 3, par exemple, est dessiné avec l'encre numéro 4. Pour le faire apparaître, il faudra donc rendre visibles toutes les encres dont le numéro contient 4 en binaire, c'est-à-dire :

4(0100),5(0101),6(0110),7(0111),12(1100),13(1101),14(1110),15(1111).

Dans une application pratique, les encres modifiées à chaque stade de l'animation seraient calculées à l'avance, si bien qu'on pourrait accélérer la partie de ce programme comprise entre les lignes **180** à **200**.

```
10 REM animation par les couleurs
20 ON BREAK GOSUB 220
30 FOR i=1 TO 15:INK i,26:NEXT
40 m(1)=1:m(2)=2:m(3)=4:m(4)=8
50 MODE 0:PRINT CHR$(23);CHR$(3);:TAG
60 FOR p=1 TO 4
70 GRAPHICS PEN m(p),1
80 LOCATE #1,1,25:PRINT #1,CHR$(48+p);
90 FOR x=0 TO 7
100 FOR y=0 TO 14 STEP 2
110 IF TEST(x*4,y)=0 THEN 140
120 MOVE (x+6)*32,(y+6)*16:PRINT CHR$(143);
130 MOVE (x+6)*32,(y+7)*16:PRINT CHR$(143);
140 NEXT y,x,p
150 LOCATE #1,1,25:PRINT#1," ";
160 FOR p=1 TO 4
170 FOR i=1 TO 25:FRAME:NEXT
180 FOR i=0 TO 15
190 IF (i AND m(p))=0 THEN INK i,0 ELSE INK i,26
200 NEXT i,p
210 GOTO 160
220 INK 1,26
run
```

Plans colorés

L'exemple précédent nous a montré comment animer à l'aide des changements de couleur des graphiques tracés avec les encres 1, 2 et 8. On peut, avec les mêmes encres et en utilisant les couleurs différemment, créer un effet complètement différent appelé « plans colorés ». En voici un exemple :

```
10 REM Montagnes
20 DEFINT a-z
30 INK 0,1:INK 1,26
40 INK 2,6:INK 3,6
50 FOR i=4 TO 7:INK i,9:NEXT
60 FOR i=8 TO 15:INK i,20:NEXT
70 MODE 0:DEG:ORIGIN 0,150:CLG:MOVE 0,150
80 FOR x=16 TO 640 STEP 16
```

```
90 DRAW x,COS(x)*150+RND*100,4
100 NEXT
110 MOVE 0,0:FILL 4
120 cx=175:GOSUB 320
130 cx=525:GOSUB 320
140 SYMBOL 252,0,0,&C,&1F,&30,&7F,&FF
150 SYMBOL 253,0,6,&E,&F2,2,&F2,&FE
160 SYMBOL 254,0,&60,&70,&7F,&7F,&7F,&7F
170 SYMBOL 255,0,0,0,&F8,&EC,&FE,&FF
180 pr$=CHR$(254)+CHR$(255)
190 pl$=CHR$(252)+CHR$(253)
200 TAG:t!=TIME
210 FOR x=-32 TO 640 STEP 4
220 x2=((608-x)*2)MOD 640:hl=RND*10:hr=50*SIN(x)
230 GRAPHICS PEN 8,1:MOVE x,100+hr,,3:PRINT pr$;
240 GRAPHICS PEN 2,1:MOVE x2,115+hl,,3:PRINT pl$;
250 IF (TEST(x2-2,115+hl-12) AND 8)=8 THEN 380
260 IF TIME-t!<30 THEN 260
270 FRAME:t!=TIME
280 GRAPHICS PEN 7,1:MOVE x,100+hr,,2:PRINT pr$;
290 GRAPHICS PEN 13,1:MOVE x2,115+hl,,2:PRINT pl$;
300 NEXT
310 GOTO 210
320 MOVE cx,100
330 FOR x=0 TO 360 STEP 10
340 DRAW cx+SIN(x)*50+10*RND,100+COS(x)*25+10*RND,1
350 NEXT
360 DRAW cx,100:MOVE cx,90:FILL 1
370 RETURN
380 ENT -1,1,1,1
390 SOUND 1,25,400,15,,1,15
400 FOR y=100+hr TO -132 STEP -2
410 GRAPHICS PEN 7,1:MOVE x,y,,2:PRINT pr$;
420 GRAPHICS PEN 8,1:MOVE x,y-2,,3:PRINT pr$;
430 NEXT
440 GOTO 70
run
```

Là encore, le fonctionnement de ce programme repose sur la forme binaire du numéro d'encre. Tous les numéros d'encre comprenant le bit « 8 » (2^3) (de 15 à 8) sont sur bleu sombre. Les numéros d'encre contenant le bit « 4 » (2^2) sont sur vert. Les encres **2** et **3**, contenant chacune le bit « 2 » (2^1) sont sur rouges, et l'encre 1 (2^0), pour finir, est sur blanc brillant, l'encre **Ø** restant bleue.

Les formes graphiques sont placées à l'écran selon la fonction OR (lignes **230** à **240**). La couleur affichée est déterminée par le bit de poids fort résultant sur chaque pixel. Une image située sur un plan à bits de poids fort est donc toujours prédominante ; l'arrière plan ne disparaît pas et redevient visible lorsque cette image est effacée. Pour effacer une image, on utilise le mode d'encre AND avec les encres 7, 11, 13 ou 14 à la place des encres 8, 4, 2 et 1 respectivement (lignes **280** et **290**).

Les graphiques exploitant la mémoire supplémentaire

Pour conclure ce chapitre, voici un programme graphique complet, « Salut l'artiste », exploitant les derniers 64 Ko de mémoire RAM.

```
10 'SALUT L'ARTISTE PAR DAVID RADISIC
20 ' copyright (c) AMSOFT 1985
30 '
40 'N'oubliez pas de faire RUN "bankman"   avant de lancer
le programme !
50 '*********************************************************
60 '
70 ON ERROR GOTO 2740
80 DEFINT a-x
90 MODE 1:ch=127:cmnd=1:pn(0)=0:pn(1)=26:pn(2)=15:pn(3)=6:pn
(4)=0:pn=1:norx=1:menu=1:zzz=HIMEM
100 DIM command$(22)
110 norx$(0)="NORMAL":norx$(1)="XOR   ":norx$(2)="transp":nor
x$(3)="XOR   "
120 RESTORE:READ cmnds$(1),cmnds$(2):cmnd$=CHR$(16)+CHR$(&7F
)+cmnds$(1)+cmnds$(2)
130 READ cmno:FOR i=1 TO cmno:READ command$(i):NEXT
140 READ st$:IF st$<>"**" THEN cmnd$(cmnd)=st$:cmnd=cmnd+1:G
OTO 140
150 WINDOW #0,1,40,1,3:PAPER #0,0:PEN #0,1:CLS #0
160 WINDOW #1,1,40,4,4:PAPER #1,3:PEN #1,1:CLS #1
170 ORIGIN 0,0,0,640,0,334
180 x=320:y=200:MOVE x,y
190 BORDER pn(4):FOR i=0 TO 3:INK i,pn(i):NEXT
200 MASK 255,0:PAPER 0:PEN 1:PAPER #1,3:PEN #1,1:GRAPHICS PE
N pn,norx
210 IF flag<>5 THEN 280
```

```
220 IF pn<2 THEN pnt$=CHR$(240):px=(pn+1)*13 ELSE IF pn<4 TH
EN pnt$=CHR$(241):px=(pn-1)*13 ELSE pnt$=CHR$(243):px=37
230 LOCATE px,2:PRINT pnt$;
240 LOCATE 1,1:PRINT USING "   pen 0 : ##   pen 1 : ##";pn(0
);pn(1);
250 LOCATE 29,2:PRINT USING "border : ##";pn(4)
260 LOCATE 1,3:PRINT USING"   pen 2 : ##   pen 3 : ##";pn(2)
;pn(3);
270 LOCATE px,2:PRINT " ";
280 LOCATE #1,1,1:PRINT #1,USING"X :####  Y  :####   ";x;y;:
PRINT #1,"MODE TRACE : ";norx$(norx+(undraw*2));" ";
290 IF flag=0 THEN GOSUB 2260
300 '
310 GOSUB 970
320 '
330 IF flag>0 THEN 390
340 IF i$="" THEN 390
350 cmnd=INSTR(cmnd$,i$):IF cmnd=0 THEN 390
360 IF cmnd=1 THEN CLG:x=320:y=200:GOTO 390
370 IF cmnd=2 THEN RUN 70
380 ON cmnd-2 GOSUB 1240,1410,1520,1640,1840,1860,1950,2020,
2090,2120,2170,2200,2660,2660,2660,2660,2390,2330,2200
390 IF tx=0 AND ty=0 THEN 200
400 IF flag >0 THEN 440
410 GOSUB 630
420 GOSUB 680:FRAME:GOSUB 680
430 GOTO 200
440 MOVE tempx,tempy,pn,1
450 ON flag GOSUB 470,490,550,640
460 GOTO 200
470 PLOT x,y:GOSUB 630:PLOT x,y
480 RETURN
490 DRAW tempx+x,tempy:DRAW tempx+x,tempy+y
500 DRAW tempx,tempy+y:DRAW tempx,tempy
510 GOSUB 630
520 DRAW tempx+x,tempy:DRAW tempx+x,tempy+y
530 DRAW tempx,tempy+y:DRAW tempx,tempy
540 RETURN
550 MOVE tempx,tempy:DRAWR x,y
560 IF triside=0 THEN 580
570 DRAW tempxx,tempyy:DRAW tempx,tempy
```

```
580 GOSUB 630
590 MOVE tempx,tempy:DRAW tempx+x,tempy+y
600 IF triside=0 THEN RETURN
610 DRAW tempxx,tempyy:DRAW tempx,tempy
620 RETURN
630 x=x+tx:y=y+ty:RETURN
640 MOVE tempx,tempy:DRAW x,y
650 GOSUB 630
660 MOVE tempx,tempy:DRAW x,y
670 RETURN
680 ' curseur trace & efface
690 IF flag=5 THEN RETURN
700 MASK 255,1
710 IF flag >1 THEN xx=tempx+x:yy=tempy+y ELSE xx=x:yy=y
720 IF flag=4 THEN xx=x:yy=y
730 IF flag=1 THEN xx=x:yy=y
740 IF undraw=1 THEN 820
750 GOSUB 790
760 MASK 255,0
770 IF i$=" "THEN GOSUB 2150:i$=""
780 RETURN
790 MOVE xx-4,yy,pn,1:DRAW xx+4,yy
800 MOVE xx,yy-4:DRAW xx,yy+4
810 MOVE xx,yy,,xorn:RETURN
820 nx=1:GOSUB 1220
830 FRAME:GOSUB 1220
840 IF i$=" " THEN nx=norx:GRAPHICS PEN pn,1:GOSUB 1220
850 i$=""
860 IF flag <>6 THEN 760
870 IF moved=0 AND j$<>"" AND (j$<CHR$(240) OR j$>CHR$(247))
 THEN ch=ASC(j$):moved=1
880 IF moved=0 THEN RETURN
890 LOCATE 5,2
900 FOR i=ch-5 TO ch+5
910 PEN ABS(i<>ch)+1
920 ch$=CHR$(1)+CHR$(ABS(i+256)MOD 256)
930 IF ch=i THEN PRINT " "ch$" ";ELSE PRINT ch$;
940 NEXT
950 PEN 1:PRINT "  = "ch"  ";
960 GOTO 760
970 ty=0:tx=0:GOSUB 680:FRAME:GOSUB 680
980 IF INKEY(0)<>-1 OR INKEY(72)<>-1 THEN ty=16
990 IF INKEY(2)<>-1 OR INKEY(73)<>-1 THEN ty=-16
1000 IF INKEY(8)<>-1 OR INKEY(74)<>-1 THEN tx=-16
```

```
1010 IF INKEY(1)<>-1 OR INKEY(75)<>-1 THEN tx=16
1020 IF INKEY(21)<>-1 OR INKEY(76)<>-1 THEN tx=tx/8:ty=ty/8
1030 IF tx=0 AND ty=0 THEN moved=0 ELSE moved=1
1040 j$=INKEY$:i$=UPPER$(j$)
1050 IF (i$=" " OR i$=CHR$(13)) AND flag>0 THEN 1090
1060 IF flag=5 THEN 1120
1070 IF flag=6 THEN 1170
1080 RETURN
1090 ON flag GOSUB 1240,1410,1640,1860,1950,2020
1100 i$=""
1110 RETURN
1120 IF moved=0 THEN RETURN
1130 IF tx>2 THEN pn=(pn+1) MOD 5 ELSE IF tx<-2 THEN pn=ABS(
(pn<1))*5-1+pn
1140 IF ty>2 THEN pn(pn)=(pn(pn)+1) MOD 27 ELSE IF ty<-2 THE
N pn(pn)=ABS((pn(pn)<1))*27-1+pn(pn)
1150 GRAPHICS PEN pn:PEN #1,pn
1160 tx=0:ty=0:BORDER pn(pn):RETURN
1170 IF tx<0 THEN ch=ABS(ch+255) MOD 256
1180 IF ty<0 THEN ch=ABS(ch+246) MOD 256
1190 IF tx>0 THEN ch=(ch+1) MOD 256
1200 IF ty>0 THEN ch=(ch+10) MOD 256
1210 tx=0:ty=0:RETURN
1220 TAG:MOVE xx-8,yy+6,pn,nx:PRINT CHR$(ch);:TAGOFF
1230 RETURN
1240 ' C
1250 IF flag=1 THEN 1290
1260 ro=1:GOSUB 2240
1270 tempx=x:tempy=y:flag=1
1280 RETURN
1290 IF tempx=x AND tempy=y THEN 1390
1300 PLOT x,y,,1
1310 tix=MAX(x,tempx)-MIN(tempx,x):tiy=MAX(y,tempy)-MIN(temp
y,y)
1320 ti=SQR((tix↑2)+(tiy↑2))
1330 ORIGIN tempx,tempy
1340 PLOT 0,0,pn,0:MOVE 0,-ti
1350 FOR z=0 TO PI*2+0.01 STEP PI/(ti/2)
1360 DRAW SIN(z+PI)*ti,COS(z+PI)*ti,pn,norx
1370 NEXT z
1380 ORIGIN 0,0
1390 x=tempx:y=tempy:tempx=0:tempy=0:flag=0
1400 RETURN
```

```
1410 ' B
1420 IF flag=2 THEN 1470
1430 ro=2:GOSUB 2240
1440 tempx=x:tempy=y:flag=2
1450 x=0:y=0
1460 RETURN
1470 IF norx=1 THEN 1500
1480 MOVE tempx,tempy:DRAW tempx+x,tempy,,norx
1490 DRAW tempx+x,tempy+y:DRAW tempx,tempy+y:DRAW tempx,temp
y
1500 x=tempx:y=tempy:flag=0
1510 RETURN
1520 ' F
1530 ro=3:GOSUB 2240
1540 GOSUB 1620:IF i$=" " THEN 1600
1550 edgecol=VAL(i$)
1560 ro=4:GOSUB 2240
1570 GOSUB 1620:IF i$=" " THEN 1600
1580 filler=VAL(i$)
1590 MOVE x,y,edgecol:FILL filler
1600 flag=0:i$=""
1610 RETURN
1620 i$=INKEY$:IF (i$<"0"OR i$>"3") AND i$<>" " THEN 1620
1630 RETURN
1640 ' T
1650 IF flag=3 THEN 1700
1660 flag=3:ro=5:GOSUB 2240
1670 tempx=x:tempy=y
1680 x=0:y=0
1690 RETURN
1700 IF triside<>0 THEN 1770
1710 ro=6:GOSUB 2240
1720 MOVE 0,0:pn,1:GOSUB 590
1730 tempxx=tempx+x:tempyy=tempy+y:x=x/2:y=20
1740 triside=1
1750 GOSUB 550:GOSUB 590
1760 RETURN
1770 IF norx=1 THEN 1800
1780 MOVE tempxx,tempyy,,norx:DRAW tempx,tempy
1790 DRAW tempx+x,tempy+y:DRAW tempxx,tempyy
1800 tempxx=0:tempyy=0
1810 x=tempx:y=tempy:triside=0
1820 tempx=0:tempy=0:flag=0
```

```
1830 RETURN
1840 ' @
1850 norx=1:undraw=undraw XOR 1:RETURN
1860 ' L
1870 IF flag=4 THEN 1910
1880 ro=7:GOSUB 2240
1890 tempx=x:tempy=y:flag=4
1900 RETURN
1910 IF norx=1 THEN 1930
1920 MOVE tempx,tempy,,norx:DRAW x,y
1930 x=tempx:y=tempy:flag=0
1940 RETURN
1950 ' I
1960 IF flag=5 THEN flag=0:CLS:INK 3,tmpcol:INK pn,col:GOTO
1990
1970 CLS:flag=5:BORDER pn(pn)
1980 RETURN
1990 FOR i=0 TO 3:INK i,pn(i):NEXT:BORDER pn(4)
2000 IF pn=4 THEN pn=1
2010 CLS:RETURN
2020 ' A
2030 IF flag=6 THEN 2070
2040 tempx=0:tempy=0:CLS
2050 undraw=1:flag=6:norx=1:moved=1
2060 RETURN
2070 flag=0
2080 RETURN
2090 ' N
2100 norx=0
2110 RETURN
2120 ' E
2130 GRAPHICS PEN pn,0:TAG:MOVE xx-8,yy+6,,0:PRINT " ";:TAGO
FF
2140 RETURN
2150 '<ESPACE>
2160 PLOT x,y,pn,norx:RETURN
2170 ' X
2180 norx=1
2190 RETURN
2200 ' M
2210 menu=menu MOD 2+1
2220 GOSUB 2260:RETURN
2230 i$=UPPER$(INKEY$):IF i$="" OR INSTR(ser$,i$)=0 THEN 223
0 ELSE RETURN
```

```
2240 CLS:undraw=0:PRINT cmnd$(ro);:LOCATE 1,3:PRINT "<espace
> ";:IF ro=3 OR ro=4 THEN PRINT "Pour Sortir"
2250 RETURN
2260 CLS:flag=-1
2270 FOR i=1 TO LEN(cmnds$(menu))
2280 ps=i+ABS(menu=2)*LEN(cmnds$(1))
2290 PEN 1:PRINT "<"MID$(cmnds$(menu),i,1)">"MID$(command$(p
s),2,4)" ";
2300 NEXT
2310 PRINT "<CLR>   <DEL>   <ESPA>";
2320 RETURN
2330 ' S
2340 GOSUB 2460:IF filename$="" THEN 2370
2350 GOSUB 2550
2360 SAVE filename$,b,&C000,&4000
2370 GOSUB 2260
2380 RETURN
2390 ' R
2400 GOSUB 2460:IF filename$="" THEN 2440
2410 GOSUB 2730
2420 LOAD filename$,&C000
2430 GOSUB 2570
2440 GOSUB 2260
2450 RETURN
2460 CLS:LOCATE 10,3:PRINT "<RETURN> pour annuler!";
2470 LOCATE 1,1:PRINT " NOM du DOSSIER :";
2480 INPUT "",filename$:IF filename$="" THEN RETURN
2490 n=INSTR(filename$,"."):IF n=0 THEN 2520
2500 IF n=1 THEN 2460
2510 filename$=LEFT$(filename$,n-1)
2520 filename$=LEFT$(filename$,8)+".scn"
2530 CLS
2540 RETURN
2550 FOR i=0 TO 4:POKE &C000+i,pn(i):NEXT
2560 RETURN
2570 FOR i=0 TO 4:pn(i)=PEEK (&C000+i) MOD 27:NEXT
2580 cn=0:FOR i=0 TO 2:IF pn(i)=pn(i+1) THEN cn=cn+1
2590 NEXT:IF cn=3 THEN 2630
2600 FOR i=0 TO 3:INK i,pn(i):NEXT
2610 BORDER pn(4):pn=1:GRAPHICS PEN pn
2620 RETURN
2630 pn(0)=0:pn(1)=26:pn(2)=15:pn(3)=6:pn(4)=0
2640 GOTO 2600
2650 ' 1, 2, 3, & 4
```

```
2660 CLS:PRINT "Desirez vous   <S>tocker":PRINT TAB(16)"<R>e
trouver":PRINT TAB(13)"ou <E>changer l'ecran?"
2670 ser$="SRE"+CHR$(13):GOSUB 2230:IF i$=CHR$(13) THEN 2260
2680 bnk2=(cmnd-13):bnk1=1
2690 IF i$="S" THEN CLS:GOSUB 2550:|SCREENCOPY,bnk2,bnk1
2700 IF i$="R" THEN GOSUB 2730:|SCREENCOPY,bnk1,bnk2:GOSUB 2
570
2710 IF i$="E" THEN CLS:GOSUB 2730:GOSUB 2550:|SCREENSWAP,bn
k2,bnk1:GOSUB 2570
2720 GOSUB 2260:RETURN
2730 FOR i=0 TO 3:INK i,0:NEXT:BORDER 0:RETURN
2740 CLS:GOSUB 2600:RESUME 2260
2750 DATA "CBFT@LIANEXM","1234RSM"
2760 DATA 19,CERCLE,"BOITE","FILL ",TRIANGLE,ALTERNE,"LIGNE"
,"INKS ",ASCII,NORMAL,EFFACER,"XOR  ","MENU ","1er  ","2eme
","3eme ",4e
me ",RETROUV,"SAUV ","MENU "
2770 DATA CERCLE,BOITE,COULEUR BORD,COULEUR DU FILL,TRIANGLE
 1,TRIANGLE 2,LIGNE,**
```

Les deux commandes RSX de ce programme, | **SCREENCOPY** et | **SCREENSWAP**, sont fournies par l'utilitaire « BANK MANAGER », stocké sur la face 1 d'une des disquettes système. Elles facilitent la copie et l'échange des écrans entre les zones de mémoire appropriées et entre les deux parties de 64 Ko de mémoire.

Vous devez donc exécuter l'utilitaire « BANK MANAGER » AVANT de lancer ce programme et, pour ce faire, insérer la face 1 d'une des disquettes système, avant de taper :

```
run "bankman"
```

Vous pouvez alors lancer « Salut l'artiste ».

Et après ?

Après lancement de BANK MANAGER, suivi de celui du programme graphique, vous voyez apparaître un menu d'options, ainsi qu'un curseur graphique, clignotant au centre de l'écran.

Il ne vous reste plus qu'à appuyer sur la lettre qui vous intéresse (C pour cercle, par exemple) pour exécuter l'option désirée.

A titre de démonstration, tapez :

```
C
```

Appuyez sur la touche fléchée vers le haut ⇧, jusqu'à ce que le curseur graphique s'élève d'environ trois centimètres au-dessus du centre de l'écran.

Appuyez enfin sur la barre d'espacement pour exécuter le cercle. Le menu réapparaît ensuite à l'écran.

Tapez **M**, pour Menu, pour passer à un menu intermédiaire qui vous permettra de <S>auvegarder, de <R>etrouver vos écrans, ainsi que de manipuler le contenu des écrans <1>, <2>, <3> et <4> (stockés dans la deuxième partie de la mémoire).

Pour exploiter ces fonctions d'écran, tapez le numéro de mémoire correspondant (**1**, **2**, **3** ou **4**). Vous obtenez alors un nouveau menu.

Il vous propose :

S	pour sauvegarder l'écran
R	pour retrouver l'écran
E	pour échanger des écrans
[RETURN]	pour arrêter l'opération en cours.

Si vous désirez, par exemple, sauvegarder l'écran courant dans la mémoire numéro 2, tapez **2** suivi de **S**.

Lorsque vous revenez au menu intermédiaire (après <R>estitution, <S>auvegarde ou manipulation d'un écran), retapez **M** pour revenir au menu de départ.

« Salut l'artiste » vous permet de dessiner des cases, des cercles, des triangles, des lignes, de placer/déplacer des points, de colorier des zones et de dessiner des caractères.

Lorsqu'un écran est terminé, il peut être sauvegardé sur disquette à l'aide de l'option <S>auvegarde du menu intermédiaire pour visualisation/modification ultérieure, et rappelé à l'écran, à tout moment, par l'option <R>etrouver.

Nous voici à la fin du dernier chapitre de ce manuel. Après utilisation, analyse et expérimentation des programmes présentés dans ce guide, vous devez maintenant avoir assimilé le BASIC Schneider et le 6128.

Pour en savoir plus ...

Vous désirez en savoir davantage sur le 6128, le BASIC, les microprogrammes, le système d'exploitation CP/M et le Dr. LOGO ? Consultez les nombreuses publications AMSOFT et les autres ouvrages disponibles sur le marché.

Pour conclure, nous vous proposons 4 annexes comprenant : le contrat d'utilisation des logiciels, un glossaire de termes, quelques programmes de jeux et un index complet du manuel.

Nous espérons que cette lecture vous a été profitable et vous remercions d'avoir fait confiance à Schneider .

Annexe 1
Contrat d'utilisation des logiciels Digital Research & Schneider

NOTICE A LIRE ATTENTIVEMENT PAR L'UTILISATEUR

N'OUVREZ PAS VOTRE PROGICIEL AVANT D'AVOIR LU CE CONTRAT. L'OUVRIR IMPLIQUE VOTRE ACCORD AVEC LES CONDITIONS ET LES LIMITES D'UTILISATION DEFINIES CI-DESSOUS.

1. Définitions

Dans ce contrat, le terme :

1. DRI signifie DIGITAL RESEARCH (CALIFORNIE) INC, propriétaire des droits d'auteur et des brevets des programmes.

2. Machine signifie le micro-ordinateur sur lequel vous utilisez le programme. Les systèmes à plusieurs micro-ordinateurs requièrent l'obtention de licences supplémentaires.

3. Programme signifie l'ensemble des programmes, de la documentation et du matériel s'y rapportant, ainsi que toutes les annexes et les mises à jour qui pourraient vous être fournies par DRI, quelle que soit l'utilisation que vous en faites ou les modifications que vous y apportez.

4. Vous assumez la pleine responsabilité du choix de votre programme, de sa mise en service, de son utilisation et des résultats qui en découleront.

2. Licence

Vous pouvez :

1. Utiliser le programme sur une seule machine.

2. Copier le programme sur support magnétique ou sur papier pour la duplication ou la modification du programme dans le cadre de l'utilisation sur une seule machine. Vous pouvez faire jusqu'à trois copies pour les raisons invoquées ci-dessus. Certains programmes pourront cependant être conçus afin de limiter ou même d'empêcher toute copie. Ils sont alors repérés en conséquence par la mention « copy protected » (protégé à la copie). La copie de la documentation est interdite. Le désassemblage est interdit.

3. Modifier le programme et/ou l'incorporer à un autre programme pour votre propre usage sur une machine unique. (Toute partie du programme incorporée à un autre logiciel continuera à être sujette aux conditions du présent contrat).

4. Céder le programme et sa licence à un tiers, à condition de notifier à DRI le nom et l'adresse de ce tiers sous réserve que ce tiers : a) accepte les conditions de licence, b) signe et envoie à DRI une copie de la carte d'enregistrement et c) paye les taxes de transfert. Si vous cédez le programme, vous devez soit en céder toutes les copies (imprimées comme magnétiques) y compris l'original, soit détruire tout ce qui n'a pas été cédé. Ceci inclut toutes les modifications et portions de programme contenues ou fusionnées dans d'autres programmes.

   Vous devez reproduire et joindre à toutes copies, modifications etc., la notice concernant les droits de reproductions réservés.

CHAQUE DISQUETTE POSSEDE UN NUMERO DE SERIE ET VOUS NE DEVEZ NI UTILISER, NI COPIER, NI MODIFIER, NI CEDER A UN TIERS LE PROGRAMME OU UNE COPIE, UNE MODIFICATION OU UNE PARTIE FUSIONNEE DU PROGRAMME, SAUF DANS LES CONDITIONS PREVUES PAR CE CONTRAT. SI VOUS CEDEZ UNE COPIE, UNE MODIFICATION OU UNE PARTIE FUSIONNEE DU PROGRAMME, VOTRE LICENCE EST AUTOMATIQUEMENT ANNULEE.

3. Echéances

Votre licence est valable jusqu'à son annulation. Vous pouvez cependant y mettre fin à tout moment en détruisant le programme ainsi que toutes ses copies, modifications et parties fusionnées. Il prend également fin sous certaines conditions citées plus loin dans ce texte, ou si vous ne respectez pas les clauses du présent contrat. Vous devez dans ce cas détruire le programme ainsi que toutes ses copies, modifications et parties fusionnées.

4. Limites de garantie

LE PROGRAMME EST LIVRE TEL QUEL. NI DRI, NI Schneider NE GARANTISSENT, SOUS QUELQUE FORME QUE CE SOIT, EXPLICITEMENT OU IMPLICITEMENT Y COMPRIS LES GARANTIES IMPLICITES A TOUT ACHAT DE MARCHANDISE, NI LE PROGRAMME, NI SON APTITUDE A ACCOMPLIR UNE TACHE PARTICULIERE. LE RISQUE, TANT DU POINT DE VUE DE LA QUALITE QUE DES PERFORMANCES DU PROGRAMME, EST ENTIEREMENT VOTRE. SI LE PROGRAMME SE REVELE DEFECTUEUX, TOUTES LES CHARGES, REPARATIONS OU CORRECTIONS SONT A VOS FRAIS.

Ni DRI, ni Schneider ne garantissent que les fonctions contenues dans ce programme vont correspondre à vos besoins ni que leur déroulement s'effectuera sans interruption ou sans erreur. Schneider garantit cependant que la disquette sur laquelle se trouve le programme est sans défaut et de bonne façon, en conditions normales d'utilisation, pour une période de 90 jours à partir de la date d'achat (attestée par une photocopie de votre reçu).

5. Limites des recours

Si votre disquette est défectueuse, Schneider s'engage à la remplacer dans les limites de garantie indiquées ci-dessus, sous réserve que vous la retourniez accompagnée de la photocopie de votre reçu.

EN AUCUN CAS, NI DRI, NI Schneider NE SERONT RESPONSABLES DES PREJUDICES CAUSES, Y COMPRIS LES PERTES EN GAIN, LES PERTES EN PROFIT OU LES DOMMAGES INDIRECTS, MEME SI DRI OU Schneider ONT ETE AVISES DE LA POSSIBILITE DE TELS PREJUDICES.

6. Carte d'enregistrement

DRI se réserve le droit de mettre à jour ses programmes de temps en temps. Les mises à jour vous seront fournies seulement dans le cas ou une carte d'enregistrement dûment signée a été retournée au siège de DRI. DRI n'est en aucun cas tenu de faire ses mises à jour, ni de vous les envoyer.

7. Généralités

Vous ne pouvez céder ni transférer la licence ou le programme, sauf spécification expresse du contrat. Tout tentative de cession de la licence, ou de transfert des droits, annule le contrat.

Le contrat est régi conformément aux termes de la loi britannique.

Pour tout renseignement, contactez DRI, à l'adresse suivante : Digital Research Inc., P.O.Box 579, Pacific Grove, California 93950.

CE CONTRAT NE POURRA ETRE MODIFIE QUE PAR UN AMENDEMENT ECRIT FAIT PAR UNE PERSONNE AUTORISEE D' Schneider OU DE DRI. TOUTE MODIFICATION APPORTEE PAR ORDRE D'ACHAT, DE PUBLICITE OU AUTRE REPRESENTATION NE SERA PAS VALABLE.

VOUS RECONNAISSEZ QUE VOUS VENEZ DE LIRE, DE COMPRENDRE ET D'AGREER AUX TERMES ET CONDITIONS DE CE CONTRAT. VOUS RECONNAISSEZ PAR AILLEURS QUE LES CLAUSES CI-PRESENTES CONSTITUENT LA TOTALITE DU CONTRAT ETABLI ENTRE VOUS, DRI ET AMSTRAD QUI ANNULE TOUTE PROPOSITION OU ACCORD PREALABLE, ORAL OU ECRIT, ENTRE VOUS, DRI ET AMSTRAD ET FAIT AUTORITE EN MATIERE DE CONTRAT.

CE CONTRAT N'INTERVIENT PAS SUR VOS DROITS STATUAIRES.

Annexe 2
Petit dictionnaire (français et anglais)

A l'intention des utilisateurs du CPC6128

Accès direct

Mode d'accès permettant de lire ou d'écrire une information en mémoire ou sur disquette dans n'importe quel ordre.

Accumulateur

Emplacement mémoire dans le microprocesseur qui retient temporairement les données à manipuler. Note : très employé en langage machine. Les utilisateurs du BASIC n'ont pas besoin de savoir qu'il existe.

Adresse

Numéro utilisé dans une instruction et qui spécifie l'emplacement d'une « cellule » dans la mémoire de l'ordinateur.

Algorithme

Nom pompeux désignant une formule ou une somme complexe ; c'est aussi une séquence d'opérations arithmétiques qui effectue une tâche précise.

Alimentation

Désigne la source d'énergie d'un système ainsi que l'électronique nécessaire à l'alimentation correcte des circuits.

Alphanumérique

Qualifie des chiffres et des lettres, par opposition aux caractères graphiques.

Amorce (Boot ou Boostrap)

Séquence d'instructions qui permet l'introduction d'un programme en mémoire ROM, initialisant ainsi le processus de chargement à un emplacement mémoire précis. Note : on parle aussi d'amorçage.

AMSDOS

Amstrad Disk Operating System : système d'exploitation Amstrad permettant au BASIC d'accèder à une ou plusieurs disquettes.

AMSOFT

Division logicielle d'Schneider qui s'occupe surtout de concevoir, d'organiser et de publier tout ce qui est nécessaire à votre 6128 et à ses nombreuses possibilités d'extension.

A/N

Voir Analogique

Analogique (Analog en anglais)

Qualifie une grandeur qui ne présente aucune discontinuité, ni dans le temps ni en amplitude. Note : l'ordinateur étant par essence un appareil numérique, il doit donc être muni de convertisseurs A/N (analogique/numérique) pour analyser des grandeurs analogiques.

Animation

Effet de « dessin animé » qui se produit lorsque l'ordinateur déplace des éléments graphiques, donnant ainsi l'illusion du mouvement.

Architecture

Organisation matérielle de l'ordinateur et de ses périphériques, qui permet d'établir les relations avec le bus de données, les échanges internes ou externes, etc.

Argument

Terme mathématique désignant une variable indépendante. Par exemple, dans l'expression $x + y = z$, x et y sont les arguments.

ASCII

Code quasiment universel qui permet de représenter les nombres, lettres et autres symboles obtenus à partir du clavier ou d'un autre type de commande.

Assembleur

Méthode pratique de programmation en code machine, où les instructions en code sont appelées par des mnémoniques (lettres qui suggèrent une fonction, ADD pour addition par exemple).

Autonome

Se dit d'un matériel lorsqu'il fonctionne indépendamment de tout autre.

Base

Première considération d'un mathématicien quand il joue avec les nombres : en binaire pour la base 2, en hexadécimal pour la base 16 et en décimal pour la base 10. Plus facile à comprendre pour les enfants que pour leurs parents.

Base de données

Ensemble de données ayant des liens entre elles, pouvant aller du simple fichier à une organisation complexe de dossiers reliés les uns aux autres.

BASIC

Langage simplifié de programmation dont l'appellation provient de l'anglais Beginners All-Purpose Symbolic Instruction Code. Note : il permet une programmation interactive et a été conçu pour un apprentissage facile. On peut essayer un programme à tout moment alors que pour les programmes compilés (voir ce mot) on doit d'abord appeler le compilateur.

BAUD

Unité de mesure du passage des données dans une transmission en série égale, la plupart du temps, au nombre de bits par seconde.

BCD (Binary Coded Decimal = Décimal Codé Binaire - DCB)

Système permettant de coder les nombres décimaux, chaque chiffre étant représenté par un groupe de 4 bits.

BDOS (Basic Disk Operating System)

Module d'exploitation des disquettes. C'est une partie du CP/M qui sert d'interface entre le programme utilisateur et les fonctions du CP/M.

Binaire (nombre)

Nombre représenté en notation binaire. Note : avec le 6128 ils sont désignés avec le préfixe &X, par exemple &X0101 est égal au nombre décimal 5.

BIOS (Basic Input Output System)

Module de gestion des entrées/sorties. Il s'agit de la partie de CP/M dépendante du matériel, spécialement écrite pour un type d'ordinateur. Toutes les entrées/sorties (écran, clavier, disquette, etc...) sont gérées par le BIOS.

Bit

Information représentée par un symbole à deux valeurs notées 0 et 1 dans le système binaire, et que l'ordinateur reproduit par des moyens électroniques.

Bit significatif

Nombre dont la signification n'apparaît qu'en binaire et dont l'influence dans un octet est prépondérante.

Bogue

Voir Bug

BOOLE (Algèbre de)

Utilisée dans les tables de vérité, où 1 veut dire vrai et 0 signifie faux.

Boot

Voir Amorce

Bootstrap

Voir Amorce

Boucle

Procédé de programmation au cours duquel l'ordinateur répète l'exécution de plusieurs lignes jusqu'à ce qu'une condition soit remplie.

Bruit

Les capacités sonores du 6128 permettent également d'introduire un nombre variable de bruits aléatoires pour créer des effets sonores, des explosions par exemple.

Bug

En Anglais c'est une bestiole. En informatique, c'est ce qui fait qu'un programme se « plante » : c'est l'erreur, le pépin, au sens le plus large. En Français, on a proposé le terme « bogue » équivalent de « bug ».

Bus

Système de connexions entre l'ordinateur, les périphériques et le monde extérieur qui transporte l'information concernant l'état du microprocesseur, la RAM et d'autres éléments matériels. Le Bus du 6128 est la prise la plus importante à l'arrière de la machine.

Byte

Voir Octet

CAD

Voir CAO

CAE

Voir EAO

Canal (Stream)

Voie de communication entre l'ordinateur et ses périphériques (imprimante, écran, disquette, cassette, etc...).

CAO (Computer Aided Design - CAD)

Conception Assistée par Ordinateur, à savoir l'ensemble des techniques informatiques utilisées dans les bureaux d'études lors du processus de conception d'un produit nouveau.

Caractère

Tout symbole qui peut être représenté : lettres, chiffres ou symboles de ponctuation et graphiques (Annexe 3)

Caractères (chaîne de)

Ensemble de caractères utilisé comme un tout pour permettre des traitements et des manipulations de texte.

Caractères (jeu de)

Toutes les lettres, nombres et symboles disponibles sur un ordinateur ou une imprimante. Si un caractère existe sur un ordinateur, cela ne veut pas obligatoirement dire qu'il peut être reçu par une imprimante quelconque.

Cartouche

Ensemble de circuits intégrés spécialement programmés se branchant sur une prise particulière de l'ordinateur. Note : Bien qu'ils soient plus onéreux que les supports magnétiques, les progiciels qu'ils contiennent se chargent plus facilement et travaillent plus rapidement que les programmes sur disquette, mais plus chers que les logiciels fournis sur disquette.

Cassette

Support de la plupart des programmes destinés aux micro-ordinateurs familiaux. Etant donné que les fichiers sont placés les uns après les autres (en ordre séquentiel), l'accès est plus lent que sur les disquettes.

Catalogue

Zone de la disquette où sont notées les entrées de chaque fichier ; une sorte de table des matières de la disquette.

CCP (Console Command Processor)

Logiciel de traitement des commandes qui permet, après que l'opérateur ait entré les commandes au clavier, de les charger et de les exécuter.

Chaîne

Souvent sous-entendu alphanumérique. Type de donnée comprenant des chiffres et des lettres ne pouvant être traitée comme une variable numérique. Peut être numérique, mais n'est considérée comme telle que si une commande le précise.

Clavier

Série de touches électroniques similaires à celles d'une machine à écrire. Note : il existe plusieurs types de clavier dont le QWERTY, d'origine anglaise, et l'AZERTY, d'origine française.

Clavier numérique

Partie du clavier regroupant des touches numériques, afin de faciliter l'entrée des données numériques, et dans le cas du 6128, des touches de fonction définissables par l'utilisateur.

Code à Barres

Code d'identification de produits, formé de barres noires plus ou moins espacées que seul un dispositif special peut lire. Note : vous le trouverez sur la plupart des emballages des produits que vous achetez (lait, lessive, biscuits, etc...)

Code Machine

Langage de programmation qui est compris par un processeur, toutes les commandes étant exprimées en binaire.

Commande

Instruction que le programme doit suivre.

Commande intégrée

Commande qui fait partie du système d'exploitation. Evitant les délais d'accès, elles sont toujours plus rapides que les commandes résidant sur disquette.

Compilateur

Programme tres complexe qui transforme un programme en langage de haut niveau tel que le BASIC, en un programme en code machine directement exécutable par la machine, donc très rapide.

Coupleur acoustique

Dispositif électronique qui comprend, entre autres, un microphone et un haut parleur sur lequel il est possible de placer un combiné téléphonique afin de transmettre des données par ligne téléphonique.

CP/M (Control Program for Microcomputers)

Programme de contrôle pour micro-ordinateurs. Note : c'est un des systèmes d'exploitation pour disquette les plus répandus, lancé par DIGITAL RESEARCH.

CPU

Central Processing Unit : alias le « chef d'orchestre » de l'ordinateur. Note : on trouve parfois UCT en français pour Unité Centrale de Traitement.

Crayon optique

Crayon qui peut parfois dessiner sur l'écran ou saisir des données, telles que les codes à barres, par exemple.

Curseur

Petit carré qui indique où va apparaître le prochain caractère sur l'écran.

Curseur graphique

Curseur semblable à un curseur de texte, mais se rapportant à l'écran graphique. Un concept invisible sur le 6128 mais indispensable pour dessiner.

Décimale (notation)

C'est notre système à base 10, avec dix chiffres de 0 à 9 facile à comprendre pour nous, mais par pour l'ordinateur, qui ne sait compter qu'en 0 et 1.

Défilement

Terme définissant le mouvement ascendant de l'affichage après remplissage de l'écran jusqu'à la dernière ligne, permettant le dégagement d'espace requis pour l'entrée ou la sortir de nouvelles informations.

Délimiteur

Voir Séparateur

Démarrage à chaud

Lorsque l'on fait un [CTRL]C sous CP/M. Le démarrage à chaud réinitialise le système d'exploitation sans détruire les données en mémoire centrale.

Démarrage à froid

Il s'agit de l'amorce et de l'initialisation d'un système d'exploitation. Le démarrage à froid du CP/M se fait par la commande | CP/M sous BASIC.

Détérioration

Destruction ou altération du contenu d'un fichier sur disquette ou de la mémoire de manière indésirable et irrécupérable.

Diagnostic

Message produit automatiquement par l'ordinateur pour indiquer et identifier une erreur dans un programme ou dans la machine elle-même.

Dialogue ordinateur-périphérique (Handshaking)

Séquence de signaux électroniques générant, vérifiant et synchronisant les échanges de données entre un ordinateur et un périphérique ou entre deux ordinateurs.

Digitaliseur (Digitizer)

Dispositif permettant de convertir des informations de type analogique en type numérique et de les transmettre à un ordinateur. Il est souvent utilisé avec les tables graphiques. Note : en français on trouve parfois le terme « numériseur ».

Disque

Plaque ayant une ou deux surfaces recouvertes d'un oxyde magnétique et qui permet de garder les données qui y sont enregistrées de manière permanente.

Données exploitables sur machine

Données ou informations directement exploitables sur ordinateur sans traitement supplémentaire à partir du clavier ou autre.

DOS (Disk Operating System = système d'exploitation sur disque)

Système qui gère l'utilisation des disques, les plus connus étant CP/M, MSDOS et UNIX.

Double face

Disquette qui peut emmagasiner des informations sur chaque face, ou unité qui permet d'avoir accès aux deux faces sans que l'on ait besoin de tourner la disquette.

Dr. LOGO

Logo de Digital Research. Il s'agit d'un langage de programmation par lequel une tortue matérialise le curseur graphique.

EAO (CAE)

Enseignement assisté par ordinateur. Ensemble des techniques et des méthodes appliquant l'informatique dans l'enseignement.

Editer

Modifier un programme, des données ou un texte.

Editeur

Programme habituellement en ROM qui permet le processus d'édition.

Editeur d'écran

Editeur de texte ou de programme permettant le déplacement du curseur en tout point de l'écran pour modifier les caractères y figurant.

Enregistrement

Ensemble d'informations formant un tout logique et physiquement liées dans les opérations de transfert entre les supports externes et la mémoire d'un ordinateur.

EPROM (Erasable Programmable Read Only Memory)

Une PROM qu'on peut programmer puis effacer à l'aide d'un rayonnement ultra-violet.

Erreur syntaxique

Erreur apparaissant lorsque les règles du programme sont contrariées par utilisation de mots clés ou de variables incorrects ; l'utilisateur en est averti par un message du BASIC.

Expression

Formule simple ou complexe qu'on utilise dans un programme pour faire un traitement sur des données ; l'expression définissant souvent le type de données à manipuler.

Fichier (File)

Fichier, programme ou groupe de données généralement stocké sur une cassette ou sur une disquette.

Firmware (Logiciel intégré, Microprogramme)

Logiciel intégré au matériel tel que le BASIC et le système d'exploitation en ROM.

Floppy disk

Mot anglais pour disquette.

Voir Disque

Formatage

Processus permettant d'organiser un système pour retrouver les fichiers et leurs références dans des secteurs particuliers de la disquette.

Forth

Langage professionnel de programmation rapide, à mi-chemin entre le haut niveau et le code machine.

Générateur de son

Partie de l'ordinateur (soit un microprocesseur soit un logiciel) qui crée les sons et les bruits.

Générations d'ordinateurs

Les avances technologiques ont marqué les étapes de l'evolution des ordinateurs et de l'informatique. La cinquième génération est en gestation, elle devrait permettre à un ordinateur de se programmer lui-même en utilisant l'Intelligence Artificielle (voir ce terme).

Graphique

Représentation plane d'une fonction mathématique : lignes, cercles, diagrammes, etc.

Graphisme de la tortue

Quand la tortue se déplace, elle laisse la trace de son passage sur l'écran graphique de Dr. LOGO.

Hexadécimal

Système de nombres à Base 16. Dans le 6128 on les précise avec les symboles & ou &H. (Par exemple &FF = décimal 255). Voir la première partie du chapitre « A vos heures de loisir... »

Horloge (Clock)

Pour la synchronisation des opérations de l'ordinateur, horloge interne chargée de donner l'heure réelle, la date, etc...

IEEE-488

Une des interfaces standard permettant de connecter des périphériques à un ordinateur.

Imprimante

Dispositif comparable à une machine à écrire automatique, branché sur l'ordinateur, afin d'obtenir sur papier les listings et les textes désirés.

Imprimante à marguerite

Imprimante de très bonne qualité de frappe et dont les caractères se trouvent sur les pétales d'une marguerite interchangeable en fonction du type d'écriture voulu. Peu rapides et bruyantes, elles sont tout de même indispensables pour obtenir un courrier de qualité.

Imprimante matricielle

Imprimante avec une grille rectangulaire de points ; la plupart des imprimantes économiques ou rapides sont de ce type. (L' Schneider DMP en est une).

Informatique

Tout ce qui touche à la manipulation des ordinateurs et de l'information dans son sens large.

Informatisation du plus grand nombre

Expression grandiloquente indiquant la mise de l'informatique à la portée de tous.

Ingénierie logicielle

Expression grandiloquente désignant la programmation informatique supposant une approche structurée et réfléchie par opposition aux techniques arbitraires.

Initialiser

Démarrer un système ou un programme en définissant les valeurs initiales des variables ou les caractéristiques initiales de son fonctionnement.

Instruction

Ordre ou commande donnés à l'ordinateur pour qu'il effectue une opération particulière. Un ensemble ou une succession d'instructions forment un programme.

Instructions (Jeu d')

Les procédés arithmétiques et logiques accomplis par le microprocesseur composent le jeu d'instructions. Une simple commande BASIC peut faire appel à des dizaines d'instructions au niveau du microprocesseur.

Intégration à grande échelle (Large Scale Integration - LSI)

Procédé technique qui permet d'intégrer sur la même puce de silicium de plus en plus de portes élémentaires de logique binaire.

Intelligence artificielle

Faculté d'un programme à apprendre et analyser une situation ou une tâche précises grâce à ses expériences passées.

Interactif

Un système est dit interactif lorsque l'ordinateur a besoin de réponses immédiates pour continuer son programme. Les jeux dépendent du mode interactif car l'action du joueur modifie la suite du déroulement du programme. On dit que l'ordinateur travaille en temps réel.

Interface

Un mot à la mode informatique pour désigner un lien (physique ou mental) entre l'ordinateur et le monde extérieur, y compris l'utilisateur, les interfaces du 6128 sont par exemple le clavier (entrée), l'écran (sortie), etc...

Interface homme-machine

Jonction entre l'ordinateur et l'opérateur : clavier, écran, son, etc...

Interface parallèle

Le 6128 en a une pour l'imprimante : plusieurs fils parallèles transmettent des informations en même temps, d'où l'adjectif « parallèle ».

Interface série

Par opposition à la précédente, ne peut transmettre qu'un bit à la fois d'une manière séquentielle ; la plus commune est appelée RS232.

Interpréteur

Par exemple, l'interpréteur du 6128 interprète le BASIC et le traduit en code machine, ligne par ligne. La « lenteur » du BASIC vient du fait qu'il traduit la même ligne plusieurs fois si elle est dans une boucle.

I/O

Input/output, l'équivalent anglais des entrées/sorties.

Itération

Une des bases de l'ordinateur, qui divise le travail en petits éléments simples qu'il répète maintes et maintes fois. C'est aussi le système des boucles.

Jeux d'Arcade et d'Aventures

Entre les envahisseurs de l'espace qui font zap, bing bang et les aventuriers qui font crac, boum, hue, il y en a pour tous les goûts. Les jeux d'aventures qui mêlent le texte, les graphismes et les contes de fées commencent à avoir leurs adeptes. Les Hobbits et le Seigneur des Anneaux y côtoient les chevaliers de la Table Ronde.

Joker

Soit *, soit ? Le Dr. LOGO n'accepte que le ? Le joker * remplace n'importe quel nombre de ? Ils sont utilisés pour bâtir des noms de fichiers ambigus. Le ? peut être n'importe quel caractère alphanumérique.

Joysticks

Alias manettes de jeux. Remplace les fonctions des touches curseur pour une action plus rapide dans les jeux.

K

Un K, ou Ko, ou encore Kilo Octet, contient 1024 Octets (2^{10}). Votre 6128 a 64K de mémoire vive, c'est-à-dire 65536 Octets (dans le genre 13 à la douzaine).

Langage évolué

Langage de programmation proche du langage humain, conçu en fonction de l'application à laquelle il est destiné et qui est pratiquement indépendant du type d'ordinateur employé.

Langage lié à l'ordinateur

Langage qui doit faire l'objet d'un assemblage afin d'être converti en code machine requis pour sa reconnaissance et son exécution par un ordinateur particulier.

Langages

Du langage machine, seul compris par l'ordinateur, au langage de haut niveau comme le BASIC, le COBOL ou le Pascal, en passant par les langages de bas niveau comme l'assembleur propre à chaque microprocesseur, l'informatique trône sur une tour de Babel où Pascal (Blaise) n'y retrouverait pas Descartes (René). Le BASIC a ses détracteurs, surtout chez les puristes, mais domine néanmoins la scène des micros.

Lecture/Ecriture (Read/Write - R/W)

Qualifie un disque, un fichier ou une unité de disquette permettant la lecture et l'écriture des données.

Lecture seulement (Read Only - R/O)

Option attribuée à une disquette, à un fichier sur disquette ou à une unité de façon à protéger les données qu'il ou qu'elle contient d'un effacement involontaire.

LISP (LIST PROCESSOR)

Nom d'un langage de programmation fonctionnelle.

Logiciel (Software)

Tout ce qui touche aux programmes et à ce qu'on met dans la machine pour la faire tourner. Sans lui, votre ordinateur n'est qu'un élément décoratif à l'esthétique futuriste.

Logiciel intégré

Voir Firmware

Logo

Nom du grec « logos » qui signifie « mot », il est utilisé dans le milieu éducatif pour l'apprentissage de la programmation.

Lutin (sprite)

Caractère pouvant se déplacer librement sur l'écran sans altérer les représentations graphiques déjà affichées.

Manette de jeux

Voir Joystick

Matériel

Ensemble des éléments physiques employés pour le traitement des données.

Matrice

Terme utilisé en mathématique et en informatique pour représenter des arrangements de variables dans plusieurs dimensions. Donne une grille de points si on l'utilise dans le sens physique. Tous les caractères sont définis sur des matrices, d'où le nom d'imprimante matricielle.

Mémoire

Un ordinateur a sa mémoire qui flanche (RAM) et sa mémoire ROM qui elle, est permanente. Bien que l'on puisse à la fois lire et écrire dans la mémoire vive RAM, ce qui s'y trouve disparaît quand on coupe le courant. On ne peut que lire ce qu'il y a dans la mémoire morte ROM. Il y a enfin la mémoire de masse, représentée par les cassettes, les disquettes, disques durs qui sont des supports amovibles de l'information mémorisée.

Mémoire (Implantation)

L'organisation de la mémoire, montrant les différentes adresses et aux fonctions l'attribution de parties précises de la mémoire, de gestion de l'écran, du système d'exploitation, etc.

Mémoire tampon (Buffer)

Dès qu'un ordinateur communique avec un appareil plus lent que lui, il a besoin de tampons pour emmagasiner les données au fur et à mesure qu'elles sont manipulées. C'est un poste de transit temporaire, sur le chemin d'une imprimante ou d'un autre périphérique.

Menu

Bien que la cuisine informatique n'ait pas l'attrait de celle de Bocuse, on utilise souvent des menus pour que l'utilisateur choisisse la substantifique moëlle. Autrement dit une liste d'options qui s'affiche sur l'écran. (Un des rares mots que les Anglo-Saxons nous aient piqués pour l'informatique).

Microprocesseur

Le cœur de l'ordinateur. Un processeur miniaturisé dont tous les éléments tiennent, en principe, en un seul circuit intégré qui exécute les instructions que lui transmet l'interpréteur BASIC.

Microprogramme

Voir Firmware

Mise au point

Elimination des défauts d'un programme qui se manifestent par des anomalies de fonctionnement.

Mode graphique

Les premiers micros avaient du mal à mélanger texte et graphiques. Le CPC664 le fait avec élégance.

Modem

Abréviation de MOdulateur DEModulateur ; on l'utilise pour les communications à distance entre ordinateurs. Cet appareil peut être à couplage acoustique ou direct. Voir aussi : coupleur acoustique.

Modulateur RF

Dispositif d'encodage et de transmission des signaux vidéo en provenance de l'ordinateur vers la prise d'antenne d'un téléviseur standard.

Moniteur

Ecran dédié d'un ordinateur, par opposition à un écran de télévision qui doit généralement être modifié ou adapté pour afficher les images venant d'un ordinateur.

Nibble

Voir Quartet

Nœuds

Unité de stockage sur l'espace de travail LOGO. Un nœud type utilise 4 octets d'espace mémoire.

Nombre à virgule fixe

Nombre dont la virgule reste à un emplacement spécifié fixe dans les représentations, les manipulations et en stockage.

Nombre réel

Nombre composé d'un nombre entier et d'une partie décimale.

Nombre sans signe

Nombre sans préfixe indiquant si sa valeur est positive ou négative.

Nombres entiers

Nombres sans partie décimale, par opposition aux réels constitués d'une partie entière et d'une partie décimale.

Nom de fichier

En LOGO un nom de fichier peut contenir huit caractères alphanumériques. En CP/M ou en AMSDOS, une extension de trois caractères précédée d'un point est permise en plus du nom proprement dit.

Nom de fichier ambigu

Nom de fichier contenant un ou plusieurs jockers. Cette ambiguité est utilisée pour travailler sur plusieurs fichiers à la fois.

Notation polonaise inversee (NPI)

Méthode de notation des opérations arithmétiques ayant la faveur de certains constructeurs de calculateurs, où les opérateurs +, -, x, / se placent derrière les valeurs auxquelles ils se rapportent.

Numeriseur

Voir Digitaliseur

Numéro de ligne

Nombres affectés aux lignes des programmes en BASIC et dans d'autres langages.

Octal

Système à base 8, où chaque chiffre (0-7) peut représenter trois bits.

Octet

Groupement de 8 bits formant la plus petite portion de mémoire que l'unité centrale puisse lire ou écrire. Le terme byte (prononcer Bait) en est la traduction anglaise.

Opérateur

Signe ou symbole (+, -, x, /) qui indique la nature d'une opération mathématique ou logique.

Organigramme

Représentation schématique de la progression des étapes d'un programme et des traitements logiques, retraçant la séquence des événements d'un programme.

Paddle

Voir Joystick

Page Zéro

Zone de mémoire centrale qui va de l'adresse &0000 à l'adresse &0100. Cette zone sous CP/M est utilisée pour stocker les très importants paramètres système.

Pas de tortue

Distance minimale parcourue par une tortue : en général un pixel.

Pascal

Langage de haut-niveau, très prisé par Acadème, qui doit être compilé avant d'être exécuté, donc très rapide. Constitue bien souvent l'étape suivante pour les étudiants du BASIC.

PEEK

Fonction BASIC qui lit le contenu d'une cellule de mémoire (octet) désignée par son adresse.

Périphérique

Tout appareil qui peut se brancher sur un ordinateur.

Périphérique logique

Représentation d'un périphérique pouvant différer de sa forme physique. Le périphérique logique du CP/M peut être associé au port Centronics ou à l'unité d'affichage VDU (Video Display Unit).

Périphérique physique

Périphérique tangible correspondant à des constituants matériels. Il peut être représenté par un périphérique logique.

Pile

Mémoire dont les données sont traitées de telle sorte que l'article extrait en premier soit le dernier article rangé, selon le principe dernier entré, premier sorti.

Pistes

Ordonnées de façon concentrique sur la disquette ; chaque piste contient le même nombre de secteurs. Les pistes et les secteurs se façonnent à un endroit spécifique au moment du formatage.

Pistes Système

Pistes réservées pour le CP/M sur une disquette.

Pixel

Plus petite partie d'une image qu'un dispositif de visualisation puisse reproduire.

POKE

Le petit frère de PEEK. Ecrivain plutôt que voyeur, il écrit une valeur à une adresse donnée.

Port

Voie d'accès qui permet d'entrer ou de recueillir des données.

Portabilité

Qualité d'un programme qui peut être exploité sur plusieurs types d'ordinateurs avec peu (ou pas) de modifications.

Porte

Circuit électronique qui emploie des composants réalisés selon une technologie particulière et assemblés selon une technique particulière. Il existe plusieurs types de portes (OU, ET, OU exclusif, etc...).

Primitive

Procédure, opération, commande résidente du Dr. LOGO.

Procédure

Ensemble des instructions à utiliser pour résoudre une tâche particulière.

Progiciel prêt à l'emploi

Expression désignant un programme s'exécutant automatiquement dès l'amorce du système. La disquette du Dr. LOGO en est un exemple.

Progiciels

Programmes d'applications spécialement développés pour une tâche particulière, prêts à l'emploi. Il en existe pour gérer l'alimentation équilibrée des animaux ou, un peu plus cher, une centrale atomique.

Programmation structurée

Technique de programmation logique et préétablie qui permet d'obtenir des programmes suivant une hiérarchie descendante (du général au particulier), selon des étapes clairement définies.

Programme

Ensemble d'instructions rédigées d'après un algorithme dans un langage accepté par l'ordinateur.

Programme d'application

A la différence d'un logiciel à caractère général, programme ayant une tâche bien précise à réaliser : un assembleur, une commande d'imprimante, etc.

Programme transitoire

Programmes utilitaires du CP/M, tel que FILECOPY, qui sont chargés en mémoire (TPA) et exécutés en entrant leur nom au clavier.

PROM (Programmable Read Only Memory)

Il s'agit d'une ROM programmable. Circuit intégré sur lequel on peut écrire une seule fois.

Prompt

Message court indiquant à l'opérateur que l'ordinateur attend quelque chose au clavier (exemple : est le prompt de CPM).

Protection en écriture

Sécurité empêchant la destruction des données sur une disquette. Une disquette protégée en écriture ne peut être que lue.

Puce (Chip)

Plaquette de semiconducteur de très petite taille, dans laquelle ont été gravés les centaines ou les milliers de transistors et de diodes qui réalisent une fonction particulière.

Quartet (Nibble)

Mot anglais signifiant petite bouchée grignotée : il représente quatre bits d'un octet, donc un demi-octet ('byte' signifie 'octet', mais également 'bouchée).

Rafraîchissement

Opération qui consiste à régénérer périodiquement les informations contenues dans une mémoire dynamique constituée de circuits qui perdent l'information au bout d'un certain temps.

RAM (Random Access Memory = Mémoire Vive - MEV)

Mémoire sur laquelle on peut écrire et lire des données, un des éléments principaux de tout micro.

RANDOM

Mot anglais qui signifie aléatoire. Il est souvent utile de pouvoir produire des nombres aléatoires. Avec le 6128 on utilise les fonctions RND et RANDOMISE. Le 6128 peut générer une séquence pseudo-aléatoire.

RASTER

Technique de balayage utilisée pour obtenir une image télévisée.

Reconnaissance de la parole

Conversion de la parole en instructions reconnaissables par l'ordinateur.

Reconnaissance optique des caractères

Action de reconnaître un caractère d'après sa forme écrite ou imprimée et de le coder par représentation binaire.

Recouvrir

Ecraser une zone de mémoire en remplaçant son contenu avec de nouvelles données.

Recursion

Fonction qui s'appelle ou se met en jeu d'elle-même.

Registre

Endroit de la mémoire du microprocesseur (CPU) qui est utilisé pour stocker temporairement des données.

REMarque

Commentaire qui ne modife en rien l'exécution du programme, bien utile pour comprendre un programme quand on ne l'a pas écrit, ou quand on l'a écrit et oublié.

Réseau

Configuration de deux ou plusieurs ordinateurs reliés pour échanger des données ou des informations par câbles ou modems.

Réservé (mot)

Mot clé qui doit être écrit sans faute et suivi d'un espace. Par exemple BASIC n'accepte pas le mot NEW en tant que variable - c'est déjà réservé.

Résolution

Capacité à distinguer des éléments d'affichage proches les uns des autres.

ROM (Read Only Memory = Mémoire Morte - MEM)

Mémoire figée ne pouvant qu'être lue, où se trouvent généralement des langages, des systèmes d'exploitation ou des programmes dont on a fréquemment besoin (cartouches ROM).

Routine (mot anglais)

Programme ou sous-programme auquel on fait souvent appel.

RS232C

Interface série la plus répandue, utilisée surtout pour les communications. Les appareils aux deux extrémités de cette interface doivent être configurés (déterminés dans leurs échanges) afin de se comprendre.

Saisie des données

Rassemblement et entrée des données sur un ordinateur.

Sauvegarde

Action de recopier en mémoire tout ou partie d'un programme, d'un texte, etc., en cours d'élaboration.

Secteur

Bloc de données. Le système d'exploitation de l'AMSTRAD gère des secteurs de 512 octets.

Séparateur

Le plus souvent, un simple espace, mais un signe d'opération peut également servir de séparateur.

Simple face

Disquette dont une seule face est disponible pour le stockage des données.

Simulation

Opération qui consiste à représenter le comportement d'un phénomène physique à l'aide de l'ordinateur.

Software

Voir Logiciel.

Sortie

Toute donnée qui provient de l'ordinateur après avoir été manipulée. Désigne aussi les interfaces qui permettent de « sortir » les données de l'ordinateur.

Sortie imprimée

Impression sur papier d'un programme, d'un texte ou d'un affichage graphique. L'équivalent sur l'écran s'appelle une visualisation.

Sous-programme

Voir Routine

Synthèse de la parole

Ensemble de techniques matérielles et logicielles visant à générer artificiellement la parole.

Table de vérité

Tableau qui indique l'état des sorties d'un circuit de logique binaire, en fonction de l'état de ses entrées.

Table graphique

Appareil qui numérise les coordonnées des points d'un dessin afin que l'ordinateur puisse les manipuler.

Tableau (Array)

Aussi appelé table, matrice ou grille. Note : il s'agit d'une organisation à une ou plusieurs dimensions. Les données sont référencées par leurs indices dans chaque dimension.

Tableur

Logiciel très prisé par les gestionnaires, appelé « spreadsheet » par les anglo-saxons, qui permet de créer des tableaux de valeurs, d'effectuer des calculs entre colonnes, de sauvegarder les résultats, etc.

Technologie de l'information

Toute technique se rapportant à l'emploi de l'électronique pour le traitement des informations et des communications : traitement de textes, communications des données, TELETEL, etc.

Temps réel

Mode de traitement qui permet l'admission des données à un instant quelconque et l'obtention immédiate des résultats.

Terminal

Autre nom d'un clavier avec ou sans moniteur qui permet d'« entrer » des données dans l'ordinateur principal.

Terminal intelligent

Terminal généralement de type conversationnel, doté de moyens informatiques propres qui le rend apte au pré- ou au post-traitement.

Terminal non-intelligent

Terminal d'ordinateur conçu pour émettre ou recevoir des données sans traitement. Un terminal « idiot » ne possède pas de dispositifs électroniques de commande d'affichage.

TPA (Transient Program Area)

Zone des programmes transitoires. Une zone mémoire qui commence à l'adresse &0100 et qui est utilisée par les programmes transitoires du CP/M.

Touche de fonction

Touche du clavier affectée à une tâche spécifique qui vient s'ajouter ou remplacer la fonction principale indiquée sur la touche.

Touche programmable

Voir touches programmables par l'utilisateur

Touches de contrôle du curseur

Touches (munies de petites flèches) qui permettent de déplacer le curseur sur l'écran, dans un sens ou dans un autre.

Touches programmables par l'utilisateur

Le 6128 en comporte 32, redéfinissables pour l'exécution de tâches multiples.

Traceur

Appareil servant à tracer automatiquement des courbes planes par mouvement d'un stylet encreur.

Traitement de texte

Type de programme ou logiciel. Indispensable dès qu'on veut écrire une lettre ou rédiger un document. Sa complexité et ses capacités varient en fonction du prix et de l'ordinateur auxquels ils sont destinés.

Transfert en aval (download)

Transfert d'information d'un ordinateur à l'autre. L'ordinateur qui reçoit les données étant celui en aval et l'ordinateur émetteur étant en amont.

Tronqué

Nombre ou chaîne réduits par suppression des caractères de début ou de fin. Lorsque la troncature est intentionnelle, la valeur peut être arrondie. Dans le cas contraire, les caractères en surplus sont tout simplement ignorés pour permettre au nombre ou à la chaîne de correspondre à l'espace disponible.

UAL (Unité Arithmétique Logique = Arithmetic Logical Unit - ALU)

Partie du microprocesseur qui effectue des opérations logiques et arithmétiques. Ceux qui programment en assembleur ou en langage machine sont directement concernées par cette notion.

UCT

Voir CPU

Unité de disquette

Dispositif qui permet d'écrire et de retrouver les données sur une disquette.

Utilitaire

Programme sur disquette qui permet à l'utilisateur d'accomplir certaines tâches (voir Programme transitoire).

Valeur par defaut

Valeur adoptée en l'absence de spécification par l'utilisateur. Sous CP/M l'unité A est l'unité par défaut.

Variable

Caractère ou groupe de caractères qui represente une valeur.

Virgule flottante

Concept informatique utilisé pour représenter les nombres de grande taille.

XYZY

Mot magique permettant de sortir des situations délicates surgissant dans les jeux d'aventure.

Annexe 3
Et pour quelques programmes de plus...

Bustout

Simple mais passionnant ! Se joue seul contre l'ordinateur, à partir du clavier ou avec les manettes.

```
10 'BUSTOUT by ALEXANDER MARTIN
20 'copyright (c) AMSOFT 1984
30 '
40 MODE 1:BORDER 1:INK 0,1:INK 1,26:INK 2,24:INK 3,6
50 SPEED KEY 15,2
60 ENV 1,1,18,0,11,0,10
70 ENT 1,10,2,2
80 ENV 3,1,0,16,5,-3,2
90 ENV 2,5,3,3,1,-21,22,9,-3,2
100 ENT -2,10,2,2,5,-7,1,2,11,3,2,-4,8
110 '
120 '
130 MOVE 30,32:DRAWR 0,400,1:MOVE 610,32:DRAWR 0,400,1
140 PEN 3:LOCATE 3,1:PRINT STRING$(36,143)
150 PEN 2:LOCATE 3,2:PRINT STRING$(36,143)
160 PEN 1:FOR r=5 TO 6:LOCATE 3,r:PRINT STRING$(36,143):NEXT
 r
170 bx=9
180 bals=5:score=0
190 PEN 1:GOSUB 680:CLEAR INPUT
200 IF INKEY$<>CHR$(32) AND JOY(0)<16 THEN 200
210 LOCATE 4,23:PRINT SPACE$(35):LOCATE 1,24:PRINT SPACE$(40
);
220 GOSUB 690:GOSUB 660:GOTO 280
230 '
240 '
250 LOCATE bx,24:PRINT "  ";STRING$(4,131);"  ":RETURN
260 '
270 '
280 xa=1:ya=1:IF INT(RND*2)=1 THEN xa=-xa
290 PEN 1:GOSUB 250
300 ORIGIN 0,400
310 x=bx+4:y=11:x1=x:y1=y
320 '
330 '
340 x1=x+xa:y1=y+ya
```



```
350 IF x1=3 OR x1=38 THEN xa=-xa
360 GOSUB 540
370 IF y1=24 AND x1>bx+1 AND x1<bx+6 THEN ya=-ya:y1=y1-2:SOU
ND 130,44,8,7,1,1:a=((x>bx+5)OR(x<bx+2)):IF a=-1 THEN xa=xa*
a:x1=x1+xa:y
1=y1+1
380 IF y1=25 THEN LOCATE x,y:PRINT " ":GOTO 500
390 GOSUB 250
400 t=TEST((16*x1)-1,-(16*y1)-1)
410 IF t<>0 THEN ya=-ya:xz=x1:yz=y1:y1=y1+ya:GOSUB 590:IF t=
2 THEN score=score+10:GOSUB 660
420 IF t=3 THEN score=score+20:GOSUB 660
430 IF t=1 THEN score=score+5:GOSUB 660
440 IF y1=1 THEN ya=1
450 LOCATE x,y:PRINT " ":LOCATE x1,y1:PRINT CHR$(233):x=x1:y
=y1
460 IF y=1 OR x=3 OR x=38 THEN SOUND 129,78,8,7,1,1
470 GOTO 340
480 '
490 '
500 bals=bals-1:SOUND 132,19,46,12,2,2:IF bals=0 THEN 620
510 GOSUB 660:GOTO 280
520 '
530 '
540 IF (INKEY(8)=0 OR INKEY(74)=0) AND bx>2 THEN bx=bx-2:RET
URN
550 IF (INKEY(1)=0 OR INKEY(75)=0) AND bx<32 THEN bx=bx+2:RE
TURN
560 RETURN
570 '
580 '
590 LOCATE xz,yz:PRINT " ":RETURN
600 '
610 '
620 IF score>hiscore THEN hiscore=score
630 GOSUB 660:score=0:vies=5:GOTO 130
640 '
650 '
660 SOUND 130,0,20,13,3,0,31: LOCATE 1,25:PRINT TAB(4)"HISCO
RE ";hiscore;
670 LOCATE 18,25:PRINT "SCORE=";score:LOCATE 30,25:PRINT "BA
LLES ";bals:RETURN
680 LOCATE 4,23:PRINT "Appuyez sur <ESPACE> pour commencer":
RETURN
690 LOCATE 1,25:PRINT SPACE$(40);:RETURN
```

Bombardier

Une version du grand classique ! Se joue seul contre l'ordinateur. A partir du clavier seulement.

```
10 ' BOMBARDIER
20 ' copyright (c)AMSOFT 1984
30 '
40 MODE 1:CLS:INK 0,0:BORDER 0:INK 1,18:INK 2,6:INK 3,4:INK
5,15:INK 6,2:INK 7,24:INK 8,8:INK 9,26:INK 10,10:INK 11,20:I
NK 12,12:INK
 13,16:INK 14,14:INK 15,21
50 SYMBOL AFTER 240:SYMBOL 241,&40,&60,&70,&7F,&7F,&3F,&7,&0
:SYMBOL 242,&0,&32,&7A,&FE,&FA,&F2,&E0,&0
60 score=0:hiscore=0:avion$=CHR$(241)+CHR$(242):x=2:y=2:chut
e=0:a=2:b=2
70 GOSUB 480
80 CLS
90 PEN 2:LOCATE 1,15:INPUT "Niveau : 0 (as) a 5 (debutant):"
,niv
100 IF niv<0 OR niv>5 THEN GOTO 90
110 niv=niv+10
120 LOCATE 1,15:PRINT CHR$(18);:LOCATE 1,15:INPUT "Vitesse :
 0 (rapide) a 100 (lent) :",vit
130 IF vit>100 OR vit<0 GOTO 120
140 '
150 ' Immeubles
160 '
170 MODE 0:FOR base=5 TO 15:FOR haut=21 TO INT(RND(1)*8+niv)
 STEP -1:LOCATE base,haut:PEN base-2:PRINT CHR$(143)+CHR$(8)
+CHR$(11)+CH
R$(244);:NEXT:NEXT
180 PLOT 0,20,4:DRAW 640,20,4
190 LOCATE 1,25:PEN 2:PRINT"score";score;:LOCATE 13,25:PRINT
"HI";hiscore;
200 '
210 ' JEU PRINCIPAL
220 '
230 LOCATE x-1,y:PRINT"   ";
240 PEN 1:LOCATE x,y:PRINT avion$;:PEN 2
250 IF y=21 AND x=15 THEN GOTO 290 ELSE GOTO 340
260 '
270 ' atterissage reussi
280 '
290 FOR c=0 TO 1000:NEXT
300 score=score+100-(niv*2):niv=niv-1:x=2:y=2:a=2:b=2:chute=
0
```



```
310 IF niv<10 THEN niv=10:vit=vit-20
320 IF vit<0 THEN vit=0
330 GOTO 150
340 FOR c=0 TO vit:NEXT
350 x=x+1
360 IF x=18 THEN LOCATE x-1,y:PRINT CHR$(18);:x=2:y=y+1:LOCA
TE x,y:PEN 1:PRINT avion$;:PEN 2
370 a$=INKEY$:IF a$=" " AND chute=0 THEN chute=1:b=y+2:a=x
380 IF y=21 THEN chute=0
390 IF chute=1 THEN LOCATE a,b:PRINT CHR$(252);:LOCATE a,b-1
:PRINT" ";:b=b+1:IF b>21 THEN LOCATE a,b:PRINT" ";:LOCATE a,
b-1:PRINT" "
;:a=0:b=0:chute=0:SOUND 3,4000,10,12,0,0,10
400 ga=(a-0.5)*32:gb=400-(b*16):bomb=TEST(ga,gb)
410 IF bomb>0 THEN GOTO 670
420 gx=((x+1.5)*32):gy=408-(y*16):crash=TEST(gx,gy)
430 IF crash>0 THEN GOTO 570
440 GOTO 230
450 '
460 '  mode d'emploi
470 '
480 LOCATE 1,2:PEN 1:PRINT" Vous pilotez un avion au-dessus
d'une  ville abandonnee que vous devez 'raser' pour atterrir
 et faire le
 plein de fuel.Votre avion se deplace  de  gauche  a  droit
e.":PRINT
490 PRINT:PRINT" Une fois le bord droit atteint,l'avion revi
ent a gauche UNE  LIGNE  PLUS  BAS. Vous disposez d'une quan
tite illimit
ee  de bombes et vous pouvez les larguer surles immeubles en
 appuyant sur la BARRE  ESPACE.";:PRINT
500 PRINT:PRINT" A chaque fois que vous atterrissez,soitla v
itesse de votre  avion  , soit  la  hauteur des immeubles au
gmente.";:PR
INT:PRINT:PRINT" VOUS NE POUVEZ LARGUER DE  BOMBE TANT  QUE
LA PRECEDANTE N'A PAS EXPLOSEE !!!";
510 PEN 2:LOCATE 1,24:PRINT:PRINT"APPUYEZ SUR UNE TOUCHE POU
R JOUER.";
520 a$=INKEY$:IF a$="" THEN GOTO 520
530 RETURN
540 '
550 ' collision
560 '
570 LOCATE x-1,y:PRINT CHR$(32)+CHR$(32)+CHR$(32)+CHR$(253)+
CHR$(8)+CHR$(238)+CHR$(8);
580 FOR t=1 TO 10 :SOUND 7,4000,5,15,0,0,5:PEN t:PRINT CHR$(
243)+CHR$(8)+CHR$(238)+CHR$(8)+CHR$(32)+CHR$(8);:NEXT:PEN 2
590 CLS:LOCATE 1,5:PRINT"SCORE:"score;
```

▶▶

```
600 IF score>hiscore THEN hiscore=score:LOCATE 1,8:PRINT"MEI
LLEUR SCORE !!";
610 score=0:LOCATE 1,12:PRINT"Tapez'R'pour rejouer"
620 a$=INKEY$:IF a$="r" OR a$="R" THEN GOTO 630 ELSE GOTO 62
0
630 PEN 1:MODE 1:x=2:y=2:a=2:b=2:GOTO 90
640 '
650 ' Immeubles Detruits
660 '
670 LOCATE a,b-1:PRINT" "+CHR$(8);:PEN 4:FOR tr=1 TO INT(RND
(1)*3)+1:score=score+5:SOUND 3,4000,10,12,0,0,10:LOCATE a,b:
FOR t=0 TO 4
:PRINT CHR$(253)+CHR$(8)+CHR$(32)+CHR$(8);:NEXT:b=b+1
680 IF b=24 THEN b=b-1
690 NEXT
700 LOCATE 6,25:PRINT score;:chute=0:a=x:b=y:GOTO 230
```

Ping-pong

C'est le plus vieux de tous les jeux, mais il fait toujours recette ! Vous pouvez jouer à deux ou seul contre l'ordinateur, à partir du clavier ou avec les manettes de jeu.

```
10 'PING PONG by DAVID RADISIC
20 'copyright (c) AMSOFT 1985
30 '
40 DEFINT a-z
50 comp=1
60 ENV 1,=11,20,=9,5000
70 MODE 1:INK 0,10:BORDER 10:INK 1,26:INK 2,18:INK 3,0
80 GOSUB 710
90 GOSUB 150
100 GOSUB 330
110 GOSUB 420
120 LOCATE 13,1:PRINT USING "#### ";score1;
130 LOCATE 35,1:PRINT USING"#### ";score2;
140 GOTO 100
150 PEN 2
160 x(1)=3:y(1)=5
170 x(2)=37:y(2)=22
180 cote$=CHR$(233):cote2$=STRING$(2,207)
190 LOCATE 1,3:PRINT STRING$ (39,cote$):PRINT STRING$(39,cot
e$)
200 FOR i=1 TO 19
210 PRINT cote2$;TAB(38);cote2$
220 NEXT
230 PRINT STRING$ (39,cote$):PRINT STRING$(39,cote$);
240 WINDOW #1,3,37,5,23
250 CLS #1
260 SYMBOL 240,0,60,126,126,126,126,60,0
270 raquette$="¦"+CHR$(8)+CHR$(10)+"¦"
280 eff$=" "+CHR$(8)+CHR$(10)+" "
290 balle$=CHR$(240)
300 PEN 3
310 LOCATE 2,1:PRINT"joueur 1 :   0";:LOCATE 24,1:PRINT"joue
ur 2 :   0";
320 RETURN
330 n=INT (RND*2):CLS #1:scored=0
340 PEN 3
350 FOR i=1 TO 2:LOCATE x(i),y(i):PRINT raquette$;:NEXT
360 ON n GOTO 390
370 xb=21:dx=1
380 GOTO 400
390 xb=19:dx=1
400 yb=12:dy=INT (RND*3)-1
```



```
410 RETURN
420 GOSUB 600
430 oxb=xb:oyb=yb
440 GOSUB 500
450 IF note>0 THEN SOUND 129,note,50,15,1
460 LOCATE oxb,oyb:PRINT" ";
470 LOCATE xb,yb:PRINT balle$
480 IF scored=0 THEN 420
490 RETURN
500 LOCATE xb+dx,yb+dy:ch$=COPYCHR$(#0)
510 note=0
520 IF ch$=" " THEN xb=xb+dx:yb=yb+dy:RETURN
530 IF ch$="¦" THEN dx=2-dx-2:dy=INT(RND*3)-1:note=200:RETUR
N
540 IF ch$=LEFT$(cote2$,1) THEN 570
550 IF ch$=cote$ THEN dy=2-dy-2:note=250
560 RETURN
570 IF dx>0 THEN score1=score1+1 ELSE score2=score2+1
580 scored=1:note=2000
590 RETURN
600 p(1)=(INKEY(69)>=0)+(INKEY(72)>=0)+ABS((INKEY(71)>=0)+(I
NKEY(73)>=0))*2
610 IF comp=1 THEN p(2)=ABS(y(2)<yb)*2+(y(2)>yb):GOTO 630
620 p(2)=(INKEY(4)>=0)+(INKEY(48)>=0)+ABS((INKEY(5)>=0)+(INK
EY(49)>=0))*2
630 PEN 3
640 FOR i=1 TO 2
650 LOCATE x(i),y(i)+p(i):ch$=COPYCHR$(#0)
660 IF ch$=" " THEN LOCATE x(i),y(i):PRINT eff$;:y(i)=y(i)+R
OUND(p(i)/2)
670 LOCATE x(i),y(i):PRINT raquette$;
680 NEXT
690 PEN 1
700 RETURN
710 PEN 2:PRINT:PRINT TAB(15)"Ping-Pong":PRINT TAB(15)"-----
----"
720 PEN 3:PRINT:PRINT TAB(14)"Pour utiliser les raquettes :"
730 PRINT:PRINT:PEN 1
740 PRINT" Joueur 1      Joueur 2        direction":PRINT
750 PRINT"    A             6                HAUT"
760 PRINT"    Z             3                BAS":PRINT
770 PEN 3:PRINT:PRINT TAB(13)"Ou les Joysticks"
780 PRINT:PRINT:PRINT:PRINT
790 PEN 2
800 PRINT TAB(6)" Choix: <1>  ou  <2>  Joueurs"
810 i$=INKEY$:IF i$<>"1" AND i$<>"2" THEN 810
820 IF i$="1" THEN comp=1 ELSE comp=0
830 MODE 1:RETURN
```

Escrime électrique

Vous devez vaincre votre adversaire ! Ne se joue qu'à deux, à partir du clavier ou avec les manettes.

```
10 'ESCRIME ELECTRIQUE par ALEXANDER MARTIN
20 'copyright (c) AMSOFT 1985
30 '
40 DEFINT a-z
50 MODE 0
60 GOSUB 980
70 GOSUB 1370
80 GOSUB 270
90 GOSUB 1520
100 GOSUB 1370
110 GOSUB 1270
120 '
130 '
140 REM   debut
150 IF fini THEN GOTO 100
160 GOSUB 240
170 FRAME:IF j1dir THEN GOSUB 570 ELSE FRAME:FRAME
180 FRAME:IF j2dir THEN GOSUB 620 ELSE FRAME:FRAME
190 IF j1sa=-1 THEN GOSUB 670
200 IF j2sa=-1 THEN GOSUB 720
210 GOTO 140
220 '
230 '
240 IF j THEN 380 ELSE 480
250 '
260 '
270 CLS:PEN 6
280 PRINT:PRINT" Choix des commandes"
290 PRINT:PRINT:PRINT:PRINT" <J>ou<C> puis ENTER"
300 LOCATE 6,10:PRINT"JOYSTICK":PRINT:PRINT"   ou CLAVIER"
310 LOCATE 15,10:IF j THEN PRINT"*"ELSE PRINT" "
320 LOCATE 15,12:IF j THEN PRINT" "ELSE PRINT"*"
330 IF NOT (INKEY(45)) THEN j=-1
340 IF NOT (INKEY(62)) THEN j=0
350 IF NOT (INKEY(18)) THEN RETURN ELSE 310
360 '
370 '
380 j1=JOY(0):j2=JOY(1)
390 j1dir=(j1 AND 1)*-1+(j1 AND 2)*0.5
400 j2dir=(j2 AND 1)*-1+(j2 AND 2)*0.5
410 IF j1 AND 16 THEN j1sa=j1sa-1:IF j1sa=-1 THEN AFTER 15 G
OSUB 770
```



```
420 IF j2 AND 16 THEN j2sa=j2sa-1:IF j2sa=-1 THEN AFTER 15 G
OSUB 770
430 IF j1sa THEN j1dir=0
440 IF j2sa THEN j2dir=0
450 RETURN
460 '
470 '
480 j2dir=((INKEY(4)=0)*1)+((INKEY(5)=0)*-1)
490 j1dir=((INKEY(69)=0)*1)+((INKEY(71)=0)*-1)
500 IF INKEY(63)=0 THEN j1sa=j1sa-1:IF j1sa=-1 THEN AFTER 15
 GOSUB 770
510 IF INKEY (10)=0 THEN j2sa=j2sa-1:IF j2sa=-1 THEN AFTER 1
5 GOSUB 770
520 IF j1sa THEN j1dir=0
530 IF j2sa THEN j2dir=0
540 RETURN
550 '
560 '
570 pt=j1wp+j1dir:IF pt>25 OR pt<6 THEN RETURN ELSE j1wp=pt
580 j1dir=0
590 PEN 1:LOCATE 3,j1wp:CLS #3:PRINT CHR$(209);:RETURN
600 '
610 '
620 pt=j2wp+j2dir:IF pt>25 OR pt<6 THEN RETURN ELSE j2wp=pt
630 j2dir=0
640 PEN 2:LOCATE 18,j2wp:CLS #5:PRINT CHR$(211);:RETURN
650 '
660 '
670 PAPER #4,4:WINDOW #4,4,17,j1wp,j1wp:CLS #4:FRAME:FRAME
680 PAPER #4,0:CLS #4
690 GOTO 570
700 '
710 '
720 PAPER #6,5:WINDOW #6,4,17,j2wp,j2wp:CLS #6:FRAME:FRAME
730 PAPER #6,0:CLS #6
740 GOTO 620
750 '
760 '
770 jwpe=(j1wp=j2wp):IF j1sa AND NOT (j2sa) AND jwpe THEN j1
sc=j1sc+1:SOUND 132,120,10,0,1,0:PRINT #1,a$(j1sc);:IF j1sc=
9 THEN 860
780 IF j2sa AND NOT (j1sa) AND jwpe THEN j2sc=j2sc+1:SOUND 1
32,100,10,0,1,0:PRINT #2,a$(j2sc);:IF j2sc=9 THEN 860
790 IF j1sa THEN SOUND 132,40,70,0,1,1
800 IF j2sa THEN SOUND 132,56,70,0,1,1
810 j1sa=0
820 j2sa=0
```



```
830 RETURN
840 '
850 '
860 PEN 6
870 LOCATE 6,10:PRINT"GAME OVER"
880 IF j1sc=9 THEN INK 1,2,20:INK 2,0 ELSE INK 2,6,17:INK 1,
0
890 SOUND 129,1000,0,12,3:SOUND 130,900,0,12,3
900 WHILE INKEY$<>"":WEND
910 t!=TIME:WHILE t!+2000>TIME:WEND
920 WHILE INKEY$="":WEND
930 CLS
940 fini=-1
950 RETURN
960 '
970 '
980  a$(0)="111101101101111"
990  a$(1)="001001001001001"
1000 a$(2)="111001111100111"
1010 a$(3)="111001111001111"
1020 a$(4)="100100101111001"
1030 a$(5)="111100111001111"
1040 a$(6)="111100111101111"
1050 a$(7)="111001001010010"
1060 a$(8)="111101111101111"
1070 a$(9)="111101111001001"
1080 FOR n=0 TO 9
1090 taille=LEN(a$(n))
1100 FOR n2=1 TO taille
1110 IF MID$(a$(n),n2,1)="1" THEN MID$(a$(n),n2,1)=CHR$(143)
 ELSE MID$(a$(n),n2,1)=CHR$(32)
1120 NEXT n2,n
1130 '
1140 '
1150 b$="ESCRIME ELECTRIQUE"
1160 c$=CHR$(32)+CHR$(164)+" Alexander Martin"
1170 ENV 1,=9,2000:ENT -1,6,3,1
1180 ENV 2,127,0,0,127,0,0,127,0,0,127,0,0,127,0,0
1190 ENV 3,=9,9000
1200 '
1210 '
1220 BORDER 0
1230 INK 0,12:PEN #4,1:PEN #6,2:PEN #1,1:PEN #2,2:PAPER #1,3
:PAPER #2,3:PEN #0,6
1240 RETURN 'FIN DE DEFINITION DES CONSTANTES
1250 '
1260 '
1270 INK 0,12:INK 1,2:INK 2,6:INK 3,13:INK 4,20:INK 5,1:INK
6,20
```

▶▶

```
1280 WINDOW #3,3,3,6,25:WINDOW #5,18,18,6,25
1290 WINDOW #1,3,5,1,5:WINDOW #2,16,18,1,5:WINDOW #7,1,20,1,
5:PAPER #7,3
1300 CLS:CLS #7:PRINT #1,a$(0);:PRINT #2,a$(0);:j1sc=0:j2sc=
0:j1wp=5:j2wp=24:j1dir=1:j2dir=1
1310 GOSUB 570:GOSUB 620
1320 SOUND 1,1000,0,12,2:SOUND 2,900,0,12,2
1330 j1sa=0:j2sa=0:fini=0
1340 RETURN 'FIN DU RENOUVELEMENT DE L'AFFICHAGE DU SCORE
1350 '
1360 '
1370 CLS
1380 PEN 7
1390 FOR n=1 TO LEN(b$)
1400 LOCATE 1+n,10
1410 FOR n2=LEN(b$) TO n STEP -1
1420 PRINT MID$(b$,n2,1)
1430 LOCATE 1+n,10
1440 SOUND 135,20*n2,5,12,2,1
1450 NEXT n2,n
1460 SOUND 135,100,0,13,3,1,20
1470 PEN 6:PRINT:PRINT:PRINT:PRINT c$
1480 FOR n=1 TO 5000:NEXT
1490 RETURN
1500 '
1510 '
1520 IF j THEN RETURN
1530 CLS
1540 LOCATE 1,5
1550 PRINT"  controle du jeu"
1560 PRINT
1570 PRINT"Joueur 1    Joueur 2"
1580 PRINT
1590 PRINT"   A    HAUT   6"
1600 PRINT"   Z    BAS    3"
1610 PRINT"   X    TIR    7"
1620 t!=TIME:WHILE t!+1000>TIME:WEND
1630 RETURN
```

DATABASE

Ce programme bien utile vous permettra de créer des petits fichiers. Vous serez de cette manière en mesure de comprendre les mécanismes utilisés pour l'ouverture et la fermeture des fichiers sur la disquette ainsi que plusieurs autres routines intéressantes (caractères double hauteur, boucles while...wend).

```
10 ' ********** DATABASE ***********
20 '
30 ' DA 12/12/84
40 ' Initialisation
50 CLEAR
60 SYMBOL AFTER 0
70 DEF FNdeek(x)=PEEK(x)+256*PEEK(x+1) ' Deek
80 SPEED WRITE 1
90 recnum = 50    ' Nombre d'enregistrements
100 ht     = 12    ' Nombre d'en-tetes
110 beep$  = CHR$(7)
120 ' Definition du code double hauteur
130 RESTORE 3520
140 double$=STRING$(81,32) : double$=""
150 FOR i=0 TO 80
160 READ j : double$=double$+CHR$(j)
170 NEXT
180 double = FNdeek(@double$+1)
190 ' definition de 6 champs/enreg.
200 DIM field1$(recnum+1),field2$(recnum+1)
210 DIM field3$(recnum+1),field4$(recnum+1)
220 DIM field5$(recnum+1),field6$(recnum+1)
230 nextblank= 1 ' pointeur sur espace pour le suivant
240 'Le libelle des champs est ds le 1er enreg.
250 RESTORE 3620 ' lit les en-tetes des champs
260 READ field1$(0),field2$(0),field3$(0)
270 READ field4$(0),field5$(0),field6$(0)
280 ' Ecran
290 MODE 1:BORDER 13
300 INK 0,13:INK 1,0:INK 2,3:INK 3,26
310 PAPER 0:PEN 1
320 GOSUB 2990 ' Illustration
330 GOSUB 370 ' Ecran d'introduction
340 GOSUB 560 ' Menu
350 END
360 ' SOUS-PROGRAMME: Ecran d'intro
370 title$="Database Amstrad"
380 GOSUB 2820
390 sp$=STRING$(40,32)
400 LOCATE 1,6
410 PRINT CHR$(24);sp$;
```



```
420 PRINT " Ce programme vous permet d'enregistrer ";sp$;
430 PRINT " jusqu'a 50 noms, avec au plus 6 champs ";sp$;
440 PRINT " par enregistrement.Le programme a ete  ";sp$;
450 PRINT " fait  pour  etre  utilise  comme  un   ";sp$;
460 PRINT " repertoire, mais il  peut etre modifie ";sp$;
470 PRINT " pour servir  de liste  de  disques,    ";sp$;
480 PRINT " de liste  de  programmes  ou  de       ";sp$;
490 PRINT " dictionnaire geographique par exemple. ";sp$;
500 PRINT CHR$(24);
510 MOVE 4,316 : DRAW 634,316,0
520 DRAW 634,50 : DRAW 4,50 :DRAW 4,316
530 GOSUB 2750 ' Appuyez sur ESPACE
540 RETURN
550 ' SOUS-PROGRAMME: Menu
560 title$="Database Amstrad": GOSUB 2820 ' en-tete
570 MOVE 10,38  : DRAW 10,330,1 : DRAW 630,330
580 DRAW 630,38 : DRAW 10,38
590 MOVE 14,42  : DRAW 14,326,1 : DRAW 626,326
600 DRAW 626,42 : DRAW 14,42
610 WINDOW 3,39,6,25
620 PRINT:PRINT "Ajouter un enreg. a database.......A"
630 PRINT:PRINT "Enlever un enreg. a database.......E"
640 PRINT:PRINT "Retrouver un enreg. particulier....R"
650 PRINT:PRINT "Changer le nom des champs..........C"
660 PRINT:PRINT "Imprimer les enreg. de database....I"
670 PRINT:PRINT "Charger un fichier.................L"
680 PRINT:PRINT "Trier les enreg. par ordre alphab..T"
690 PRINT:PRINT "Sauvegarder un fichier.............S"
700 WINDOW 1,40,1,25
710 WHILE G$>"" : G$=INKEY$ : WEND
720 WHILE G$="" : G$=INKEY$ : WEND
730 option = INSTR("AETCRLSI",UPPER$(G$))
740 ON option GOSUB 810,970,1170,2400,1550,1850,2060,2260
750 '  OPTIONS :     A    E    T    C    R    L    S    I
760 IF option THEN 560 ' commande a effectuer
770 PRINT beep$; : GOTO 710
780 '
790 ' SOUS-PROGRAMME: Ajoute un enreg.
800 '
810 title$="Ajout d'un enrg" : GOSUB 2820 ' en-tete
820 IF nextblank > recnum THEN LOCATE 14,8 : PRINT "Database
 plein" : GOTO 2750
830 PRINT"C'est l'enreg. numero";nextblank
840 PRINT
850 PRINT field1$(0);TAB(ht);:INPUT ":",field1$(nextblank)
```

▶▶

```
860 PRINT field2$(0);TAB(ht);:INPUT ":",field2$(nextblank)
870 PRINT field3$(0);TAB(ht);:INPUT ":",field3$(nextblank)
880 PRINT field4$(0);TAB(ht);:INPUT ":",field4$(nextblank)
890 PRINT field5$(0);TAB(ht);:INPUT ":",field5$(nextblank)
900 PRINT field6$(0);TAB(ht);:INPUT ":",field6$(nextblank)
910 nextblank=nextblank+1
920 GOSUB 2750
930 RETURN
940 '
950 ' SOUS-PROGRAMME: Efface un enreg.
960 '
970 title$="Effacer un enrg." : GOSUB 2820 ' en-tete
980 INPUT "Effacer quel enrg." ; dl
990 PRINT
1000 IF dl >= nextblank THEN PRINT "Cet enreg. n'existe pas"
 : GOTO 1120
1010 d=dl : st=0 : GOSUB 2640 ' affiche l'enreg.
1020 INPUT "O pour confirmer l'effacage:",q$
1030 PRINT
1040 IF UPPER$ (q$)<>"O" THEN PRINT TAB(12);"Pas d'effacage"
: GOTO 1120
1050 FOR i=d+1 TO nextblank
1060 field1$(i-1)=field1$(i):field2$(i-1)=field2$(i)
1070 field3$(i-1)=field3$(i):field4$(i-1)=field4$(i)
1080 field5$(i-1)=field5$(i):field6$(i-1)=field6$(i)
1090 NEXT
1100 nextblank=nextblank-1
1110 PRINT TAB(18);"Effacage"
1120 GOSUB 2750 ' espace
1130 RETURN
1140 '
1150 ' SOUS-PROGRAME: tri
1160 '
1170 title$="Tri  enreg." : GOSUB 2820 ' en-tete
1180 PRINT : PRINT TAB(8);"Tri des enrg. par ";field1$(0)

1190 ' Shell/Metzner No. 1 Classement des field1$ (1 a num%)
1200 ' debute avec num%=nombre d'enreg. : utilise l'integral
e
1210 timethen = TIME ' chronometre le classement
1220 num%=nextblank-1
1230 s1%=num%
1240 WHILE s1% >1
1250     s1%=s1% / 2
1260     s2%=num%-s1%
1270     flag%=1
1280     WHILE flag%=1
1290         flag%=0
1300         FOR s3%=1 TO s2%
```



```
1310            s4%=s1%+s3%
1320            IF field1$(s3%) > field1$(s4%) THEN GOSUB 1440
1330            NEXT s3%
1340        WEND
1350    WEND
1360 timenow =  TIME - timethen
1370 timenow =  INT(timenow/30) / 10 ' en seconde a la dizai
ne pres.
1380 LOCATE 6,12 : PRINT "Classement en";timenow;"secondes"
1390 GOSUB 2750 ' espace
1400 RETURN ' au menu principal
1410 '
1420 ' SOUS-PROGRAMME: Permutation
1430 '
1440 s1$=field1$(s3%):field1$(s3%)=field1$(s4%):field1$(s4%)
 = s1$
1450 s1$=field2$(s3%):field2$(s3%)=field2$(s4%):field2$(s4%)
 = s1$
1460 s1$=field3$(s3%):field3$(s3%)=field3$(s4%):field3$(s4%)
 = s1$
1470 s1$=field4$(s3%):field4$(s3%)=field4$(s4%):field4$(s4%)
 = s1$
1480 s1$=field5$(s3%):field5$(s3%)=field5$(s4%):field5$(s4%)
 = s1$
1490 s1$=field6$(s3%):field6$(s3%)=field6$(s4%):field6$(s4%)
 = s1$
1500 flag%=1
1510 RETURN ' au sous-prg de Classement
1520 '
1530 ' SOUS-PROGRAMME: Recherche
1540 '
1550 title$="Trouver un enrg." : GOSUB 2820 ' en-tete
1560 LOCATE 3,6
1570 PRINT"pour tout lire appuyez sur ENTER"
1580 LOCATE 1,8
1590 INPUT "Lettres a rechercher :",qu$
1600 qu$=UPPER$(qu$)
1610 FOR se=1 TO nextblank-1
1620 q$ = qu$
1630 IF INSTR( UPPER$(field1$(se)) , q$ ) THEN GOSUB 1750
1640 IF INSTR( UPPER$(field2$(se)) , q$ ) THEN GOSUB 1750
1650 IF INSTR( UPPER$(field3$(se)) , q$ ) THEN GOSUB 1750
1660 IF INSTR( UPPER$(field4$(se)) , q$ ) THEN GOSUB 1750
1670 IF INSTR( UPPER$(field5$(se)) , q$ ) THEN GOSUB 1750
1680 IF INSTR( UPPER$(field6$(se)) , q$ ) THEN GOSUB 1750
1690 NEXT
```

▶▶

```
1700 CLS : GOSUB 2750 ' espace
1710 RETURN ' au menu principal
1720 '
1730 ' SOUS-PROGRAMME: Affiche l'enreg. trouve
1740 '
1750 CLS
1760 PRINT : PRINT
1770 d=se : st=0
1780 GOSUB 2640 ' Affiche l'enreg.
1790 GOSUB 2750 ' espace
1800 q$=CHR$(0) ' pour eviter d'ecrire plusieurs fois le mem
e item
1810 RETURN
1820 '
1830 ' SOUS-PROGRAMME : Chargement fichier
1840 '
1850 title$="Charger un fichier": GOSUB 2820 ' en-tete
1860 PRINT"Mettez la disquette contenant le "
1870 PRINT"fichier dans le lecteur puis"
1880 PRINT"entrez le nom du fichier"
1890 PRINT:INPUT "Nom: ",f$
1900 PRINT:OPENIN f$
1910 INPUT #9,nextblank
1920 FOR i=0 TO nextblank-1
1930 INPUT #9,field1$(i)
1940 INPUT #9,field2$(i)
1950 INPUT #9,field3$(i)
1960 INPUT #9,field4$(i)
1970 INPUT #9,field5$(i)
1980 INPUT #9,field6$(i)
1990 NEXT i
2000 CLOSEIN
2010 GOSUB 2750 ' espace
2020 RETURN
2030 '
2040 ' SOUS-PROGRAMME: Sauvegarder 1 fichier
2050 '
2060 title$="Sauvegarde des enreg.": GOSUB 2820 ' en-tete
2070 PRINT"Placez une disquette dans le lecteur"
2080 PRINT"Entrez le nom du fichier :"
2090 PRINT:INPUT "Nom: ",f$
2100 PRINT:OPENOUT f$
2110 PRINT #9,nextblank
2120 FOR i=0 TO nextblank-1
2130 PRINT #9,field1$(i)
2140 PRINT #9,field2$(i)
```



```
2150 PRINT #9,field3$(i)
2160 PRINT #9,field4$(i)
2170 PRINT #9,field5$(i)
2180 PRINT #9,field6$(i)
2190 NEXT
2200 CLOSEOUT
2210 GOSUB 2750 ' espace
2220 RETURN
2230 '
2240 ' SOUS-PROGRAMME:Impression
2250 '
2260 title$="Imprimer les enreg." : GOSUB 2820 ' en-tete
2270 INPUT "Avez vous une imprimante connectee (O/N)",q$
2280 IF UPPER$(q$)<>"O" THEN CLS : st=0 ELSE st=8
2290 PRINT #st,"DATABASE AMSTRAD"
2300 PRINT #st : PRINT #st
2310 FOR d=0 TO nextblank-1
2320 GOSUB 2640 ' affiche enreg.
2330 NEXT
2340 PRINT #st : PRINT #st
2350 GOSUB 2750 ' espace
2360 RETURN
2370 '
2380 ' SOUS-PROGRAMME: Nom champs
2390 '
2400 title$="Nom des champs" : GOSUB 2820 ' en-tete
2410 PRINT : PRINT "en-tete 1:";field1$(0)
2420 INPUT "Change en:",fi$
2430 IF fi$>"" THEN field1$(0)=fi$
2440 PRINT : PRINT "en-tete 2:";field2$(0)
2450 INPUT "Change en:",fi$
2460 IF fi$>"" THEN field2$(0)=fi$
2470 PRINT : PRINT "en-tete 3:";field3$(0)
2480 INPUT "Change en:",fi$
2490 IF fi$>"" THEN field3$(0)=fi$
2500 PRINT : PRINT "en-tete 4:";field4$(0)
2510 INPUT "Change en:",fi$
2520 IF fi$>"" THEN field4$(0)=fi$
2530 PRINT : PRINT "en-tete 5:";field5$(0)
2540 INPUT "Change en:",fi$
2550 IF fi$>"" THEN field5$(0)=fi$
2560 PRINT : PRINT "en-tete 6:";field6$(0)
2570 INPUT "Change en:",fi$
2580 IF fi$>"" THEN field6$(0)=fi$
2590 GOSUB 2750 ' espace
```

▶▶

```
2600 RETURN
2610 '
2620 ' SOUS-PROGRAMME: Affiche L'enreg. en cours
2630 '
2640 PRINT #st,"L'enreg.";d;"est:" : PRINT #st
2650 PRINT #st,field1$(0);TAB(ht);":";field1$(d)
2660 PRINT #st,field2$(0);TAB(ht);":";field2$(d)
2670 PRINT #st,field3$(0);TAB(ht);":";field3$(d)
2680 PRINT #st,field4$(0);TAB(ht);":";field4$(d)
2690 PRINT #st,field5$(0);TAB(ht);":";field5$(d)
2700 PRINT #st,field6$(0);TAB(ht);":";field6$(d)
2710 PRINT #st : RETURN
2720 '
2730 ' SOUS-PROGRAMME: Espace
2740 '
2750 LOCATE 10,25
2760 PRINT "ESPACE pour continuer";
2770 WHILE INKEY$<>" " : WEND
2780 RETURN
2790 '
2800 ' SOUS-PROGRAMME: Affichage du titre
2810 '
2820 tl=LEN(title$)+2
2830 tt=INT((40-tl)/2)
2840 CLS
2850 PAPER 1 : PEN 3
2860 LOCATE tt,1 : PRINT STRING$(tl,32)
2870 LOCATE tt,2 : PRINT" ";title$;" "
2880 LOCATE tt,3 : PRINT STRING$(tl,32)
2890 PAPER 0 : PEN 1
2900 PRINT ' pour effacer une ligne a la fin
2910 tx=tt*16-12 : tx1=(tt+tl)*16-20
2920 MOVE tx,396 : DRAW tx1,396,0
2930 DRAW tx1,354 : DRAW tx,354
2940 DRAW tx,396
2950 RETURN
2960 '
2970 ' SOUS-PROGRAMME: Image ecran
2980 '
2990 CLG 0:Z%=16
3000 RESTORE 3460
3010 Y%=250:FOR X%=400 TO 10 STEP -Z%*2
3020 Y%=Y%-Z%
3030 READ N$
3040 GOSUB 3310
```

```
3050 NEXT
3060 ORIGIN 0,0,0,639,0,399
3070 FOR X%=630 TO 616 STEP -1:MOVE X%,338:DRAW X%,X%-296,2:
NEXT
3080 FOR X%=0 TO 16 STEP 2:MOVE X%,140:DRAW X%,140+X%/1.5,2:
NEXT
3090 ORIGIN 0,0,0,234,30,140 : CLG 2
3100 ORIGIN 0,0,0,639,0,399
3110 Y%=30:FOR X%=234 TO 630 STEP 2
3120 MOVE X%,Y%:DRAW X%,Y%+110,2
3130 Y%=Y%+1 : NEXT
3140 MOVE 16,152:DRAW 0,140,3
3150 DRAW 0,30:DRAW 234,30
3160 DRAW 630,226:DRAW 630,340
3170 DRAW 616,340:MOVE 630,338
3180 DRAW 234,140:DRAW 0,140
3190 MOVE 234,140:DRAW 234,30
3200 LOCATE 2,2 : X$="Database" : GOSUB 3400 ' Double
3210 LOCATE 2,5 : X$="Amstrad": GOSUB 3400 ' Double
3220 LOCATE 27,20 : X$="Appuyez sur" : GOSUB 3400 ' Double
3230 LOCATE 27,23 : X$="une touche" : GOSUB 3400 ' Double
3240 WHILE INKEY$="" : WEND
3250 RETURN
3260 '
3270 ' SOUS-PROGRAMME: Dessine une carte en couleur 3
3280 ' ligne couleur 1 contient
3290 ' reference N$,N (4 chiffres)
3300 '
3310 ORIGIN 0,0,X%,X%+208,Y%,Y%+140
3320 CLG 1:MOVE X%,Y%:DRAWR 0,140,1:DRAWR 214,0:DRAWR 0,-140
:DRAWR -214,0
3330 MOVE X%+6,Y%+4:DRAWR 202,0,0:DRAWR 0,132:DRAWR -202,0:D
RAWR 0,-132:MOVER 0,112:DRAWR 202,0
3340 MOVER -194,16
3350 TAG:PRINT N$;:TAGOFF
3360 RETURN
3370 '
3380 ' SOUS-PROGAMME: Affiche X$ en double hauteur
3390 '
3400 FOR i=1 TO LEN(x$)
3410 POKE double+1,ASC(MID$(X$,i,1))
3420 CALL double : NEXT : RETURN
3430 '
3440 ' Noms pour l'image ecran
3450 '
3460 DATA "Willams:7471","Stevens:6216","Soames :5807","Smit
h  :2201"
```

▶▶

```
3470 DATA "Nesbit :2207","Marks  :2022","Lipton :1091","Gree
n  :8087"
3480 DATA "Frinton:1011","Evans  :2877","Barker :8123","Alto
n  :9981"
3490 DATA "  DATABASE"
3500 '
3510 ' Code machine pour le texte double hauteur
3520 DATA &3E,&6E,&CD,&A5,&BB,&16,&FE,&3E,&19
3530 DATA &CD,&5A,&BB,&7A,&CD,&5A,&BB,&0E,&04,&7E
3540 DATA &CD,&5A,&BB,&CD,&5A,&BB,&23,&0D,&20,&F5
3550 DATA &14,&20,&E7,&3E,&FE,&CD,&5A,&BB,&3E,&0A
3560 DATA &CD,&5A,&BB,&3E,&08,&CD,&5A,&BB,&3E,&FF
3570 DATA &CD,&5A,&BB,&3E,&0B,&CD,&5A,&BB,&C9,&5A
3580 DATA &BB,&3E,&0A,&CD,&5A,&BB,&3E,&08,&CD,&5A
3590 DATA &BB,&3E,&FF,&CD,&5A,&BB,&3E,&0B,&CD,&5A
3600 DATA &BB,&C9
3610 ' Nom de champs par defaut
3620 DATA Nom,Rue,Ville,Pays,Code Postal,Tel.No
```

▶▶

Arsène Lupin

Il s'agit de s'introduire dans la maison de Sa Seigneurie et d'y dérober le butin. Vous devez franchir une multitude d'obstacles et éviter le chien ! Se joue seul contre l'ordinateur, à partir du clavier ou avec les manettes.

```
10 'ARSENE LUPIN by DAVID RADISIC
20 'copyright (c) AMSOFT 1985
30 '
40 MODE 0:INK 0,0:BORDER 0:INK 1,26:INK 2,15:INK 3,25
50 INK 4,14:INK 5,24:INK 6,0:INK 7,0:INK 8,0:PAPER #1,7
60 delais=200
70 DIM objx(5,20),objy(5,20),gemx(5,20),gemy(5,20)
80 GOSUB 380
90 GOSUB 720
100 pause=200:GOSUB 340
110 IF gems=0 THEN GOSUB 970
120 PEN 4
130 FOR i=10 TO 12
140 LOCATE 15,i:PRINT"BUTIN";
150 NEXT
160 PAPER 0:CLS #2:PAPER 8
170 GOSUB 1170
180 GOSUB 1230
190 GOSUB 1370
200 GOSUB 1510
210 IF rm=0 THEN GOSUB 1900
220 IF mort=0 THEN 160
230 pause=100:GOSUB 340
240 PAPER 0:CLS:PEN 1
250 LOCATE 5,3:PRINT"VOULEZ-VOUS";
260 LOCATE 7,5:PRINT"REJOUER";
270 PEN 5:LOCATE 9,7:PRINT"O/N";
280 i$=UPPER$(INKEY$):IF i$<>"O" AND i$<>"N" THEN 280
290 IF i$="N" THEN MODE 2:PEN 1:STOP
300 RUN
310 IF chien=1 THEN RETURN
320 chien=1:chienx=minx(rm):chieny=miny(rm)
330 RETURN
340 FOR loop=1 TO pause
350 FRAME
360 NEXT
370 RETURN
380 rm=1:xp=6:yp=4:homme$=CHR$(224):chien=0:vol=0
390 SYMBOL 240,8,8,8,8,8,8,8,8
400 SYMBOL 241,0,0,0,0,255,0,0,0
410 SYMBOL 242,0,0,0,0,15,8,8,8
420 SYMBOL 243,0,0,0,0,248,8,8,8
430 SYMBOL 244,8,8,8,8,248,0,0,0
```



```
440 SYMBOL 245,8,8,8,8,15,0,0,0
450 SYMBOL 246,8,12,13,14,12,12,8,8
460 SYMBOL 247,8,12,12,14,13,12,8,8
470 SYMBOL 248,8,24,88,56,24,24,8,8
480 SYMBOL 249,8,24,24,56,88,24,8,8
490 SYMBOL 250,0,0,255,255,255,255,255,0
500 SYMBOL 251,28,20,20,20,20,20,20,28
510 SYMBOL 252,0,0,255,255,255,255,255,0
520 SYMBOL 253,28,28,28,28,28,28,28,28
530 SYMBOL 255,195,165,60,126,90,60,36,24
540 ENT 1,12,-4,1
550 ENT -2,=1000,60,=3000,40
560 ENV 1,10,1,5,2,-4,1,2,-1,20
570 fenet$(1)=STRING$(2,250):fenet$(2)=CHR$(251)+CHR$(8)+CHR
$(251)+CHR$(8)+CHR$(10)+CHR$(251)
580 porte$(1)=STRING$(2,252):porte$(2)=CHR$(253)+CHR$(8)+CHR
$(10)+CHR$(253)+CHR$(8)+CHR$(10)+CHR$(253)
590 inter$(1,0)=CHR$(246):inter$(1,1)=CHR$(247)
600 inter$(2,0)=CHR$(248):inter$(2,1)=CHR$(249)
610 gem$=CHR$(144):obj$=CHR$(233):chien$=CHR$(255)
620 coup$=CHR$(246)+CHR$(248)+CHR$(247)+CHR$(249)+CHR$(252)+
CHR$(253)+CHR$(250)+CHR$(251)+gem$+obj$+chien$
630 RESTORE 3010
640 FOR i=1 TO 5
650 READ minx(i),miny(i),maxx(i),maxy(i)
660 READ dir(i,1),dir(i,2),dir(i,3),dir(i,4)
670 NEXT
680 WINDOW #1,minx(rm)-1,maxx(rm)+1,miny(rm)-1,maxy(i)+1
690 WINDOW #2,1,14,1,25
700 CLS #1:PAPER #0,8
710 RETURN
720 ORIGIN 50,50
730 INK 6,24,12
740 RESTORE 3060
750 GOSUB 1280
760 LOCATE 2,20
770 PEN 5:PRINT"--";
780 PEN 1:PRINT"echappatoires";
790 PEN 5:PRINT"--";
800 LOCATE 8,2:PRINT"ENTREE";
810 pause=300:GOSUB 340
820 CLS:LOCATE 1,3:INK 6,0
830 PEN 1:PRINT homme$;"   A.LUPIN (VOUS)":PRINT
840 PEN 2:PRINT LEFT$(porte$(1),1);LEFT$(porte$(2),1);"
   Portes":PRINT
850 PEN 3:PRINT inter$(1,0);inter$(2,0);"interrupteur eteint
"
860 PEN 3:PRINT inter$(1,1);inter$(2,1);"interrupteur allume
":PRINT
870 PEN 4:PRINT LEFT$(fenet$(1),1);LEFT$(fenet$(2),1);"
   Fenetres":PRINT
```

▶▶

```
880 PEN 5:PRINT gem$;"          Bijoux":PRINT
890 PAPER 1:PEN 0:PRINT obj$;"          Obstacles":PEN 1:PAPE
R 0:PRINT
900 PEN 1:PRINT chien$;"          Le chien"
910 PEN 5:PRINT:PRINT
920 PRINT"Utilisez le Joystick":PRINT"   Ou les touches":PRI
NT"      flechees"
930 feint=REMAIN(1)
940 AFTER delais*4,1 GOSUB 340
950 RETURN
960 '
970 'GENERATEUR DE RUBIS/OBSTACLES
980 '
990 FOR piece=1 TO 5
1000 gemr=INT(RND*8)+2:objr=INT(RND*10)+5
1010 minx=minx(piece):miny=miny(piece):maxx=maxx(piece):maxy
=maxy(piece)
1020 FOR i=1 TO gemr
1030 x=INT(RND*(maxx-minx+1))+minx
1040 y=INT(RND*(maxy-miny+1))+miny
1050 gemx(piece,i)=x:gemy(piece,i)=y
1060 gems=gems+1
1070 NEXT i
1080 FOR i=1 TO objr
1090 x=INT(RND*(maxx-minx+1))+minx
1100 y=INT(RND*(maxy-miny+1))+miny
1110 objx(piece,i)=x:objy(piece,i)=y
1120 NEXT i
1130 gems(piece)=gemr:obj(piece)=objr
1140 NEXT piece
1150 CLS
1160 RETURN
1170 ON rm GOTO 1180,1190,1200,1210,1220
1180 RESTORE 2670:RETURN
1190 RESTORE 2740:RETURN
1200 RESTORE 2810:RETURN
1210 RESTORE 2880:RETURN
1220 RESTORE 2960:RETURN
1230 PAPER 0:READ rm$:PAPER 8
1240 WINDOW #1,minx(rm)-1,maxx(rm)+1,miny(rm)-1,maxy(rm)+1:C
LS #1
1250 PEN 1:LOCATE 1,1:PRINT SPACE$(19);
1260 LOCATE 1,1:PRINT"Piece :";rm$;
1270 IF lumiere(rm) THEN INK 7,10:INK 8,10 ELSE INK 7,0:INK
8,0
1280 READ a$:IF a$="END" THEN RETURN
1290 IF a$="D" THEN 2180
1300 IF a$="W" THEN 2260
1310 IF a$="L" THEN GRAPHICS PEN 1:GOTO 2340
1320 IF a$="S" THEN 2420
1330 IF a$="F" THEN GRAPHICS PEN 6:GOTO 2340
```

▶▶

```
1340 PRINT"*** ERREUR ***";
1350 STOP
1360 '
1370 ' AFFICHAGE BIJOUX/OBJETS
1380 '
1390 PEN 6
1400 FOR i=1 TO obj(rm)
1410 LOCATE objx(rm,i),objy(rm,i)
1420 PRINT obj$;
1430 NEXT
1440 PEN 5
1450 FOR i=1 TO gems(rm)
1460 LOCATE gemx(rm,i),gemy(rm,i)
1470 PRINT gem$;
1480 NEXT
1490 PEN 1:LOCATE xp,yp:PRINT homme$;
1500 RETURN
1510 xf=0:yf=0:PEN 1
1520 IF INKEY(0)<>-1 OR INKEY(72)<>-1 THEN yf=-1
1530 IF INKEY(2)<>-1 OR INKEY(73)<>-1 THEN yf=1
1540 IF INKEY(8)<>-1 OR INKEY(74)<>-1 THEN xf=-1
1550 IF INKEY(1)<>-1 OR INKEY(75)<>-1 THEN xf=1
1560 IF xf=0 AND yf=0 THEN 1630
1570 LOCATE xp+xf,yp+yf:ht$=COPYCHR$(#0)
1580 IF ASC(ht$)>239 AND ASC(ht$)<246 THEN 1510
1590 IF ht$<>" " THEN 1660
1600 LOCATE xp,yp:PRINT" ";
1610 PAPER 0:LOCATE 15,5:PRINT"       ";:PAPER 8
1620 xp=xp+xf:yp=yp+yf
1630 LOCATE xp,yp:PRINT homme$;
1640 IF chien>0 THEN chien=chien MOD 2+1:IF chien=2 THEN 255
0
1650 GOTO 1510
1660 coup=INSTR(coup$,ht$):char=ASC(MID$(coup$,coup,1))
1670 ON coup GOTO 1690,1690,1690,1690,1750,1750,1850,1900,19
70,2090,2650
1680 GOTO 1600
1690 IF coup>2 AND coup<5 THEN char=char-1
1700 IF coup<3 THEN char=char+1
1710 PEN 3:LOCATE xp+xf,yp+yf:PRINT CHR$(char);
1720 lumiere(rm)=lumiere(rm) XOR 1
1730 IF lumiere(rm) THEN INK 7,10:INK 8,10 ELSE INK 7,0:INK
8,0
1740 GOTO 1510
1750 IF xf<>0 AND yf<>0 THEN 1630
1760 IF xf<0 THEN dir=4 ELSE IF xf>0 THEN dir=3
1770 IF yf<0 THEN dir=1 ELSE IF yf>0 THEN dir=2
1780 IF dir(rm,dir)=-1 THEN 1630 ELSE rm=dir(rm,dir)
1790 IF chien>0 THEN GOSUB 310
1800 IF dir=1 THEN xp=6:yp=maxy(rm)
```

▶▶

```
1810 IF dir=2 THEN xp=6:yp=miny(rm)
1820 IF dir=3 THEN xp=minx(rm):yp=13
1830 IF dir=4 THEN xp=maxx(rm):yp=13
1840 RETURN
1850 IF xp>5 AND xp<8 THEN 1880
1860 IF xp<6 THEN dir=4 ELSE dir=3
1870 GOTO 1780
1880 IF yp>13 THEN dir=2 ELSE dir=1
1890 GOTO 1780
1900 PAPER 0:CLS:PEN 1
1910 LOCATE 3,3:PRINT"vous vous en tirez !";
1920 LOCATE 9,5:PRINT"avec";
1930 IF gems=vol THEN LOCATE 8,7:PRINT"TOUT !"; ELSE LOCATE
9,7
1940 PRINT USING " ##";vol;
1950 PEN 5:LOCATE 8,9:PRINT "Bijoux";
1960 mort=1:RETURN
1970 LOCATE xp,yp:PRINT " ";:xp=xp+xf:yp=yp+yf
1980 i=0
1990 i=i+1
2000 IF i>gems(rm) THEN 1510
2010 IF gemx(rm,i)<>xp OR gemy(rm,i)<>yp THEN 1990
2020 IF i=gems(rm) THEN 2050
2030 gemx(rm,i)=gemx(rm,gems(rm))
2040 gemy(rm,i)=gemy(rm,gems(rm))
2050 gems(rm)=gems(rm)-1:vol=vol+1
2060 MOVE 400,150+(vol*2),1,1:DRAW 560,150+(vol*2),1,1
2070 SOUND 129,248,10,12,0,1
2080 GOTO 1980
2090 bruit=INT(RND*15)
2100 SOUND 1,3000,10,bruit,0,0,10
2110 PAPER 0:LOCATE 15,5:PRINT"CRACK!";:PAPER 8
2120 IF bruit<10 OR delais=50 THEN 1630
2130 delais=delais-50
2140 feint=REMAIN(1)
2150 AFTER delais*4,1 GOSUB 310
2160 GOTO 1630
2170 '
2180 ' dessin des portes
2190 '
2200 READ no,dr$
2210 IF dr$="V" THEN dr=2 ELSE dr=1
2220 PEN 2
```



```
2230 pic$=porte$(dr):GOSUB 2500
2240 GOTO 1280
2250 '
2260 ' dessin des fenetres
2270 '
2280 READ no,wi$
2290 IF wi$="V" THEN wi=2 ELSE wi=1
2300 PEN 4
2310 pic$=fenet$(wi):GOSUB 2500
2320 GOTO 1280
2330 '
2340 ' tire les traits
2350 '
2360 READ x1,y1,x2,y2
2370 MOVE x1,y1,,0
2380 DRAW x1,y2,,0:DRAW x2,y2,,0
2390 DRAW x2,y1,,0:DRAW x1,y1,,0
2400 GOTO 1280
2410 '
2420 ' dessine les interrupteurs
2430 '
2440 READ no,sw$
2450 IF sw$="L" THEN sw=1 ELSE sw=2
2460 PEN 3
2470 pic$=inter$(sw,0):GOSUB 2500
2480 GOTO 1280
2490 '
2500 ' affiche le caractere
2510 '
2520 READ x,y:LOCATE x,y:PRINT pic$;
2530 no=no-1:IF no>0 THEN 2520
2540 RETURN
2550 PEN 1:LOCATE chienx,chieny:PRINT" ";
2560 homme$=CHR$(225)
2570 IF (chienx=xp AND chieny=yp) OR (chienx=xp+xf AND chien
y=yp+yf) THEN 2650
2580 IF chienx<xp THEN chienx=chienx+1
2590 IF chienx>xp THEN chienx=chienx-1
2600 IF chieny<yp THEN chieny=chieny+1
2610 IF chieny>yp THEN chieny=chieny-1
2620 LOCATE chienx,chieny:PRINT chien$;
2630 SOUND 1,0,RND*40,10,1,2,31
2640 GOTO 1510
2650 PRINT"SNAP!";
```

▶▶

```
2660 mort=1:RETURN
2670 DATA ENTREE
2680 DATA L,64,308,226,4
2690 DATA D,2,H,6,3,6,22
2700 DATA D,2,V,4,12,9,11
2710 DATA S,1,L,4,11
2720 DATA S,1,R,9,14
2730 DATA END
2740 DATA SALON
2750 DATA L,2,308,258,4
2760 DATA D,1,V,10,12
2770 DATA W,1,H,6,3
2780 DATA W,1,V,2,12
2790 DATA S,2,R,10,11,10,15
2800 DATA END
2810 DATA SALLE A MANGER
2820 DATA L,2,308,258,4
2830 DATA W,1,V,10,12
2840 DATA W,1,H,6,3
2850 DATA D,1,V,2,12
2860 DATA S,2,L,2,11,2,15
2870 DATA END
2880 DATA CUISINE
2890 DATA L,2,276,384,4
2900 DATA D,2,H,6,5,6,22
2910 DATA W,1,H,10,22
2920 DATA W,1,V,14,13
2930 DATA D,1,V,2,13
2940 DATA S,1,L,2,16
2950 DATA END
2960 DATA LINGERIE
2970 DATA L,2,276,256,4
2980 DATA D,1,V,10,12
2990 DATA S,1,R,10,11
3000 DATA END
3010 DATA 5,4,8,21,0,4,3,2
3020 DATA 3,4,9,21,-1,-1,1,-1
3030 DATA 3,4,9,21,-1,-1,-1,1
3040 DATA 3,6,13,21,1,0,-1,5
3050 DATA 3,6,9,21,-1,-1,4,-1
3060 DATA L,64,308,480,100
3070 DATA F,250,98,294,102
3080 DATA F,250,306,294,310
3090 DATA F,390,94,430,106
```



```
3100 DATA F,390,302,430,314
3110 DATA F,474,240,488,270
3120 DATA F,474,124,488,154
3130 DATA F,58,240,72,270
3140 DATA L,226,308,322,180
3150 DATA L,160,180,480,100
3160 DATA L,64,180,160,100
3170 DATA END
```

Annexe 4
Index

(Le premier numéro renvoie au chapitre et le second à la page. Ainsi, pour la référence 1.44 reportez-vous au chapitre 1, page 44.)

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

U

V

W

X

Y

Z

Imprimé en Allemagne · Holzmann-Druck-Service · D-8939 Bad Wörishofen

Schneider
COMPUTER DIVISION

GUIDE de L'UTILISATEUR

Colour Personal Computer CPC6128